TUGAS AKHIR – TE 145561

RANCANG BANGUN *MASTER CLOCK* DENGAN SINKRONISASI GPS BERBASIS ARDUINO

Candra Mashuri
NRP. 2214030068
Samsul Hidayatulloh
NRP. 2214030079

Dosen Pembimbing
Ir. Gatot Kusrahardjo, MT.
Suwito, ST., MT.

PROGRAM STUDI KOMPUTER KONTROL
DEPARTEMEN TEKNIK ELEKTRO OTOMASI
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

FINAL PROJECT – **TE 145561**

DESIGN BUILD MASTER CLOCK WITH ARDUINO BASED GPS SYNC

Candra Mashuri
NRP. 2214030068
Samsul Hidayatulloh
NRP. 2214030079

Supervisor
Ir. Gatot Kusrahardjo, MT.
Suwito, ST., MT.

COMPUTER CONTROL STUDY PROGRAM
DEPARTEMENT OF ELECTRICAL ENGINEERING AUTOMATION
Faculty of Vocational
Sepuluh Nopember Insitute of Technology
Surabaya 2017

PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul :

"Rancang Bangun *Master Clock* dengan Sinkronisasi GPS Berbasis Arduino"

adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, 19 Juli 2017

Candra Mashuri
NRP. 2214030068

Surabaya, 19 Juli 2017

Samsul Hidayatulloh
NRP. 2214030079

Halaman ini sengaja dikosongkan

RANCANG BANGUN *MASTER CLOCK* DENGAN SINKRONISASI *GPS* BERBASIS ARDUINO

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan Untuk
Memperoleh Gelar Ahli Madya Teknik Elektro
Pada
Bidang Studi Computer Control
Jurusan D3Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui

Dosen Pembimbing I,	Dosen Pembimbing II,
Ir. Gatot Kusrahardjo, MT.	Suwito, ST., MT.
NIP. 19590428 198601 1 001	NIP. 19810105 200501 1 004

SURABAYA
JULI, 2017

Halaman ini sengaja dikosongkan

RANCANG BANGUN *MASTER CLOCK* DENGAN SINKRONISASI GPS BERBASIS ARDUINO

Nama : Candra Mashuri
NRP : 2214 030 068
Nama : Samsul Hidayatulloh
NRP : 2214 030 079

Pembimbing I : Ir. Gatot Kusrahardjo, MT.
NIP : 195904281986011001
Pembimbing II : Suwito, ST., MT.
NIP : 198101052005011004

ABSTRAK

Pada sistem jaringan berbasis komputer, setiap perangkat yang tehubung dalam satu jaringan menggunakan komunikasi berupa data digital. Apabila *clock* pada masing-masing perangkat tidak sesuai maka komunikasi data pada perangkat akan terjadi kesalahan. Hal ini menyebabkan ketidak sesauian antar perangkat sehingga perangkat tersebut tidak berjalan semestinya.

Pada proyek tugas akhir ini dengan melakukan penelitian untuk membuat *master clock* yang tersinkronisasi dengan GPS untuk membuat *clock* yang terhubung pada *master clock* menjadi sama. Penambahan fitur pada *master clock* yaitu fitur untuk membunyikan suara tiap jam dan aplikasi *timer* yang dikendalikan menggunakan *smartphone.*

Hasil dari proyek tugas akhir ini setelah melakukan 4 kali pengujian di tempat berbeda maka waktu untuk sinkronisasi dengan GPS paling cepat pada tempat terbuka dengan waktu 16 detik. Untuk perangkat yang terhubung pada *master clock* dengan jarak 60 m *clock* pada perangkat sama dengan *clock* pada *master clock.*

Kata Kunci : *masterclock*,GPS, *sound, smartphone.*

Halaman ini sengaja dikosongkan

DESIGN MASTER CLOCK BUILD WITH ARDUINO BASED GPS SYNC

Name	**: Candra Mashuri**
Register Number	**: 2214 030 068**
Name	**: Samsul Hidayatulloh**
Register Number	**: 2214 030 078**
Supervisor I	**: Ir. Gatot Kusrahardjo, MT.**
ID Number	**: 19590428 1986011001**
Supervisor II	**: Suwito,ST,. MT.**
ID Number	**: 198101052005011004**

ABSTRACT

On a computer-based network system, each device connected in a network uses digital data communications. If the clock on each device does not match then the data communication on the device will occur error. This causes a mismatch between devices so the device is not running properly.

In this final project by doing research to create a synchronized master clock with GPS to make the clock connected to the master clock to be the same. The addition of features on the master clock is a feature to sound every hour and timer applications are controlled using a smartphone.

The result of this final project after performing 4 tests in different places then the time to sync with the fastest GPS in the open place with a time of 16 seconds. For devices connected to the master clock with a distance of 60 m the clock on the device is equal to the clock in the master clock.

Keywords: *master clock, GPS, sound, smartphone.*

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji dan syukur penulis panjatkan kehadirat Allah SWT, karena berkat rahmat dan karunia-Nya

Tugas Akhir ini disusun untuk memenuhi sebagian persyaratan guna menyelesaikan pendidikan Diploma 3 pada Program Studi Komputer Kontrol, Departemen Teknik Elektro Otomasi, Fakultas Vokasi, Institut Teknologi Sepuluh Nopember Surabaya dengan judul:

Rancang Bangun *Master Clock* dengan Sinkronisasi GPS Berbasis Arduino

Laporan penelitian ini dapat diselesaikan oleh penulis berkat bantuan, bimbingan, dan dukungan dari berbagai pihak. Penulis ingin mengucapkan terima kasih kepada Ibu, Bapak, dan Keluarga tercinta yang selalu memberi dukungan, semangat, dan doa untuk keberhasilan penulis. Bapak Ir. Gatot Kusrahardjo, MT. dan Bapak Suwito, ST., MT. selaku dosen pembimbing atas bimbingan dan arahannya. Bapak Joko Subur ,ST., MT terima kasih karena telah banyak membantu kami menyelesaikan tugas akhir ini. Teman-teman dan semua pihak yang tidak dapat penulis sebutkan satu persatu.

Penulis berharap laporan ini dapat bermanfaat bagi pembaca pada umumnya dan penulis pada khususnya. Laporan ini masih jauh dari sempurna, sehingga penulis mengharapkan kritik dan saran dari pembaca yang bersifat membangun.

Surabaya, 19 Juli 2017

Penulis

Halaman ini sengaja dikosongkan

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi yang semakin pesat menuntut setiap perangkat untuk saling terhubung baik dalam jangkauan ruangan atau global. Komunikasi dari tiap perangkat yang berbasis komputer menggunakan komunikasi data digital, apabila dari komunikasi tersebut hilang atau tidak sesuai dapat mengganggu suatu kinerja dari perangkat yang saling terhubung dalam suatu jaringan. Ketidak sesuaian dari komunikasi data antar perangkat disebabkan dari penerimaan data digital yang *clock* nya berbeda. Ini menyebabkan perangkat yang menerima data digital tersebut jadi salah membaca data yang diterima.

Pada spesifikasi sistem waktu menggunakan sistem *clock*. Sistem *clock* mempunyai dua fungsi utama yaitu *clock generation* dan *clock distribution. Clock generation* digunakan untuk sinyal dengan akurasi tinggi, setelah itu disinkronkan dengan suatu sistem. Pada gelombang pulsa, sinyal sinkronisasi disebut *clock.* Pada *clock distribution* gelombang sinyal yang tersinkronisasi akan distribusikan ke semua alat yang terhubung pada satu jaringan. [1] Kesibukan dunia kerja membuat orang banyak lupa dengan waktu mengakibatkan waktu yang dihabiskan tidak sesuai dengann yang diharapkan.

Oleh karena itu pada proyek tugas akhir ini dengan menerapkan sistem *master clock* yang digunakan untuk mensinkronisasikan dari beberapa alat berupa jam digital dengan *clock* dari *GPS (Global Positioning System)*. GPS akan mengambil data *clock* dari *NTP (Network Time Protocol)* lalu data *clock* disinkronisaikan dengan RTC oleh *masterclock* untuk perangkat jam digital. Pengembangan dari *masterclock* yaitu penambahan *sound* pada *masterclock* yang digunakan untuk menyuarakan waktu tiap jam. Penggunaan *sound* pada *masterclock* berguna untuk membantu tuna netra atau orang yang sibuk kerja dan lupa waktu untuk mengetahui waktu melalui suara dari tiap jam. Penambahan fitur lain dari *master clock* yaitu dengan *clock* pada perangkat yang dapat diatur dengan memanfaatkan *smartphone.*

1.2 Perumusan Masalah

Komunikasi data digital dari tiap perangkat berbasis komputer yang terhubung dalam satu jaringan mempunyai *error* yang disebabkan karena clock dari tiap perangkat tidak sesuai. Sehingga sistem tersebut tidak berjalan sesuai dengan yang diinginkan. *Clock* dari tiap perangkat harus sinkron agar sistem dapat berjalan sesuai dengan yang diinginkan. Pada proyek tugas akhir ini dirancang alat berupa *masterclock* agar perangkat-perangkat yang dikendalikan dapat sinkron.

1.3 Tujuan

Tujuan proyek tugas akhir ini memiliki 2 tujuan, tujuan yang pertama dilakukan oleh Candra Mashuri dan tujuan kedua dilakukan oleh Samsul Hidayatulloh, yaitu bertujuan:

1. Merancang sistem *master clock* untuk menangkap data dari GPS lalu disinkronisasi dengan RTC agar *set* waktu pada RTC sesuai dengan UTC.
2. Merancang sistem komunikasi pada *master clock* dengan *slave clock* dan penambahan fitur *timer* yang dapat dikendalikan dengan *smartphone* dan fitur suara yang dapat membunyikan waktu tiap jam.

1.4 Batasan Masalah

Dari perumusan masalah di atas, ada beberapa hal yang perlu dibatasi sehingga penelitian yang dilakukan dapat tercapai.Batasan masalah dalam pengerjaan Tugas Akhir ini, yaitu:

1. Penelitian ini merupakan modifikasi dari penelitian sebelumnya dan dikerjakan secara berkelompok dengan kesepakatan perbedaan titik kerja anggota kelompok.
2. Pengambilan data waktu pada GPS untuk *reset* RTC setiap hari.
3. Komunikasi antara *master clock* dengan *slave clock* menggunakan kabel.
4. Fitur suara untuk membunyikan waktu tiap jam.
5. Aplikasi *timer* pada *slave clock* yang dikendalikan dengan *smartphone*.

1.5 Sistematika Laporan

Pembahasan Tugas Akhir ini akan dibagi menjadi lima Bab dengan sistematika sebagai berikut:

Bab I **Pendahuluan**

Bab ini meliputi latar belakang, permasalahan, tujuan penelitian, metodologi penelitian, sistematika laporan, dan relevansi.

Bab II **Teori Dasar**

Bab ini menjelaskan tentang tinjauan pustaka yang menunjang dalam pengerjaan alat ini. Bab ini dibagi menjadi dua bagian dimana Candra Mashuri mencari bahan dari materi pada poin 2.1 sampai dengan poin 2.4. dan samsul Hidayatulloh mencari bahan dari materi pada poin 2.5 sampai dengan poin 2.8.

Bab III **Perancangan Sistem**

Pada bab ini akan dibahas mengenai peracangan sistem dari alat yang akan dibuat. Candra Mashuri bertugas untuk merancang *hardware* dan *software* untuk sinkronisasi GPS dengan *master clock* dan *reset* RTC menggunakan data waktu GPS. Samsul Hidayatulloh bertugas untuk merancang dan membuat *hardware* dan *software* yang berhubungan komunikasi antara *master clock* dan *slave clock, timer*.

Bab IV **Simulasi, Implementasi dan Uji Coba**

Bab ini akan dijelaskan mengenai uji coba pada masing masing komponen dan juga data dari komponen. Pada bab ini Candra Mashuri bertangung jawab pada pegujian yang berhubungan dengan sinkronisasi GPS dan RTC pada *master clock*. Begitu juga dengan Samsul Hidayatulloh bertanggung jawab pada pengujian yang berhubungan dengan *slave clock, timer* dan suara tiap jam. Tiap poin dijelaskan siapa yang bertanggung jawab

Bab V **Penutup**

Bab ini berisi kesimpulan dan saran dari hasil pembahasan yang telah diperoleh.

1.6 Relevansi

Semua perangkat berbasis komputer dalam satu jaringan akan tersinkronisasi dengan kontrol dari *masterclock*. Sebagai alat bantu penunjuk waktu bagi tuna netra dan orang sibuk karena *sound* menyuarakan waktu tiap jam. Semua perangkat berbasis komputer yang terhubung pada *master clock* dapat diatur *clock* nya dengan memanfaatkan *smartphone*.

BAB II
TEORI DASAR

Pada bab ini akan dibahas mengenai tinjauan pustaka dimana ada 9 poin. Tinjauan pustaka ini dibagi menjadi 2 bagian tiap masing masing mahasiswa, Candra Mashuri mencari bahan dari materi pada poin 2.1 sampai dengan poin 2.4. dan Samsul Hidayatulloh mencari bahan dari materi pada poin 2.5 sampai dengan poin 2.8. dimana tiap materi yang dicari sebagai dasar materi untuk pengerjaan alat yang dibuat masing-masing mahasiswa, dan juga sebagai dasar materi untuk pembuatan keselurahan alat ini.

2.1 *Master Clock* [2]

Master clock adalah jam presisi yang menyediakan sinyal waktu untuk menyinkronkan *slave clock* sebagai bagian dari jaringan jam. Jaringan jam listrik yang dihubungkan oleh kabel ke jam pendulum *master* presisi mulai digunakan di institusi seperti pabrik, kantor, dan sekolah sekitar tahun 1900. Saat ini banyak jam radio disinkronkan oleh sinyal radio atau koneksi internet ke sistem waktu di seluruh dunia yang disebut *Coordinated Universal Time* (UTC) yang diatur oleh jam atom utama di banyak negara.

Jam sinkron berguna karena berbagai alasan. Seringkali sistem terdistribusi adalah dirancang untuk mewujudkan beberapa perilaku yang disinkronkan, terutama dalam pengolahan *real time* di pabrik, pesawat terbang, kendaraan luar angkasa, dan aplikasi militer. Jika jam disinkronkan, algoritma dapat dilanjutkan dalam "putaran:" dan algoritma yang dirancang untuk sistem sinkron dapat digunakan. Dalam sistem database, manajemen versi dan kontrol konkurensi bergantung pada kemampuan menetapkan cap waktu dan nomor versi ke *file* atau entitas lainnya Beberapa algoritma yang menggunakan *timeout*, seperti komunikasi protokol, sangat tergantung waktu [3].

2.2 GPS BR 355 [4]

GPS BR 355 dapat digunakan dalam berbagai aplikasi yang memerlukan sinyal GPS untuk diterima dan diterjemahkan dalam laptop. Kompatibel dengan hampir semua perangkat lunak pihak ketiga yang sesuai dengan NMEA, GPS BR 355 sangat sesuai untuk digunakan dalam navigasi kendaraan, laut dan penerbangan, serta aplikasi komersial dan

pemerintahan seperti truk pemadam kebakaran, mobil polisi, bus, pengumpulan data GIS dan masih banyak lagi.

BR 355 mempunyai bentuk yang ramping dan tahan air. Dilengkapi antena yang aktif untuk tingkat akurasi GPS tertinggi. BR 355 memerlukan kabel USB atau *Serial* RS232 / PS2 agar berfungsi.Penempatan penerima GPS di manapun di dalam kendaraan biasanya akan mendapatkan penerimaan sinyal GPS yang memadai. Gambar GPS BR355 ada pada Gambar 2.1.

Gambar 2.1. GPS BR 355

2.3 RTC DS 3231 [5]

Modul RTC DS 3231 (*Real Time Clock*) memiliki akurasi dan presisi yang sangat tinggi dalam mencacah waktu dengan menggunakan IC RTC DS 3231. DS 3231 memiliki kristal internal dan rangkaian kapasitor *tuning* di mana suhu dari kristal dimonitor secara berkesinambungan dan kapasitor disetel secara otomatis untuk menjaga kestabilan detak frekuensi. Pencacahan waktu pada solusi RTC lain dapat bergeser hingga hitungan menit per bulannya, terutama pada kondisi perubahan suhu yang ekstrim.

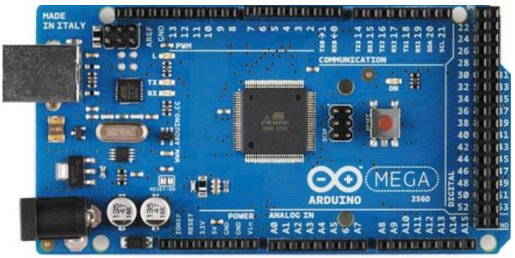
Modul ini paling jauh hanya bergeser kurang dari 1 menit per tahunnya, dengan demikian modul ini cocok untuk aplikasi kritis yang sensitif terhadap akurasi waktu yang tidak perlu disinkronisasikan secara teratur terhadap jam eksternal.Akses modul ini dilakukan melalui antarmuka I2C yang dapat dikatakan identik dengan pengalamatan *register* pada RTC DS 1307, dengan demikian kode program yang sudah dibuat untuk Arduino atau mikrokontroler lain dapat berjalan tanpa perlu dimodifikasi. Modul ini juga sudah dilengkapi dengan IC AT24C32 yang

memberikan EEPROM tambahan sebesar 4 KB (32,768 bit) yang dapat digunakan untuk berbagai keperluan, misalnya untuk menyimpan jadwal (*time schedule*), menyimpan setelan waktu alarm, menyimpan data hari libur pada kalender, merekam absensi, dsb. Alamat dari EEPROM ini dapat disetel dengan menghubung-singkatkan pada A0, A1, dan A2 (8 pilihan alamat). Gambar modul RTC DS 3231 ada pada Gambar 2.2

Gambar 2.2. Modul RTC DS 3231

2.4 Arduino Mega [6]

Arduino Mega adalah *board* Arduino yang merupakan perbaikan dari *board* Arduino Mega sebelumnya. Arduino Mega awalnya memakai *chip* ATmega 1280 dan kemudian diganti dengan *chip* ATmega 2560, oleh karena itu namanya diganti menjadi Arduino Mega. Arduino jenis ini merupakan papan mikrokontroler yang berbasis ATmega 2560. Terdapat 54 pin *input/output* digital (14 diantaranya dapat digunakan sebagai *output* PWM), 16 *input analog*, sebuah *crystal osilator* 16MHz, sebuah koneksi USB, sebuah *jack power*, sebuah *header* ICSP, dan sebuah tombol *reset*. Arduino Mega seperti ditunjukkan pada Gambar 2.3.



Gambar 2.3. Arduino Mega

2.5 Arduino Uno [7]

Arduino Uno adalah *board* mikrokontroler berbasis ATmega328 (*datasheet*). Memiliki 14 pin *input* dari *digital output* dimana 6 pin *input* tersebut dapat digunakan sebagai *output* PWM dan 6 pin *analog input*, 16 MHz osilator kristal, koneksi USB, *jack power*, ICSP *header*, dan tombol *reset*. Untuk mendukung mikrokontroler agar dapat digunakan, cukup hanya menghubungkan *board* Arduino Uno ke komputer dengan menggunakan kabel USB atau listrik dengan AC yang ke adaptor DC atau baterai untuk menjalankannya.

Uno berbeda dengan semua *board* sebelumnya dalam hal koneksi USB-*to-serial* yaitu menggunakan fitur Atmega8U2 yang diprogram sebagai konverter USB-*to-serial* berbeda dengan *board* sebelumnya yang menggunakan *chip* FTDI *driver* USB-*to-serial*. Arduino uno seperti ditunjukkan pada Gambar 2. 4.

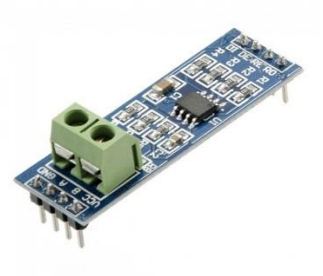
Gambar 2.4. Arduino Uno

2.6 RS 485 [8]

IC RS 485 adalah IC yang digunakan untuk teknik komunikasi data *serial.* Sistem komunikasi dengan menggunakan RS 485 mulai dikembangkan pada tahun 1983 di mana dengan teknik ini, komunikasi data dapat dilakukan pada jarak yang cukup jauh yaitu 1,2 Km. Selain dapat digunakan untuk jarak yang jauh teknik ini juga dapat digunakan untuk menghubungkan 32 unit beban sekaligus hanya dengan menggunakan dua buah kabel saja tanpa memerlukan referensi ground yang sama antara unit yang satu dengan unit lainnya. RS-485 merupakan standar komunikasi *serial* yang bersifat *multidrop/multi-point*. Dalam sistem *multipoint* ini *transfer* data dapat dilakukan dari satu *transmitter* ke beberapa *receiver* sekaligus, atau dengan kata lain membentuk suatu

jaringan komputer. Dalam RS 485 terdapat sebuah *transmitter* (disebut juga *driver*) dan sebuah *receiver* .[9]

Keistimewaan RS 485 ini antara lain terletak pada transmisi diferensialnya (sering disebut juga sebagai *balanced transmission*). Dalam transmisi diferensial ini level tegangan TTL diterjemahkan menjadi selisih tegangan antara *output* A dan B. Dengan demikian efek dari *noise* dapat diminimalkan, karena interferensi *noise* akan terjadi sekaligus pada jalur *output* (A) dan jalur *complementary output* (B) sehingga selisih tegangan antara *output* A dan B tetap. Modul RS terlihat seperti pada Gambar 2.5.



Gambar 2.5. Modul RS 485

2.7 Modul *Bluetooth* HC 05 [10]

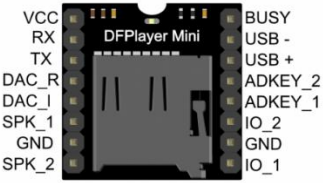
Modul *bluetooth* HC 05 merupakan modul komunikasi nirkabel pada frekuensi 2,4GHz dengan pilihan koneksi bisa sebagai *slave*, ataupun sebagai *master*. Sangat mudah digunakan dengan mikrokontroler untuk membuat aplikasi *wireless*. *Interface* yang digunakan adalah *serial* RXD, TXD, VCC dan GND. Tampilan LED sebagai indikator koneksi *bluetooth.*

Tegangan *input* antara 3,6V sampai 6V, jangan menghubungkan dengan sumber daya lebih dari 7V. Arus saat tidak terhubung sekitar 30mA, dan saat terhubung sebesar 10mA. 4 pin *interface* 3,3V dapat langsung dihubungkan ke berbagai macam mikrokontroler (khusus Arduino, 8051, 8535, AVR, PIC, ARM, MSP430, dsb). Jarak efektif jangkauan sebesar 10 m, meskipun dapat mencapai lebih dari 10 m, namun kualitas koneksi makin berkurang.

Gambar 2.6. Modul HC 05 *Bluetooth*

2.8 *DFPalyer Mini* [11]

DFPlayer Mini untuk arduino adalah modul MP3 dengan ukuran kecil dan harga terjangkau dengan pengaturan *output* yang mudah untuk *speaker*. Modul ini bisa berdiri sendiri dengan *battery*, *speaker*, dan *push button* atau bisa dikombinasikan dengan arduino. Modul *DFPlayer Mini* terlihat pada Gambar 2.7.

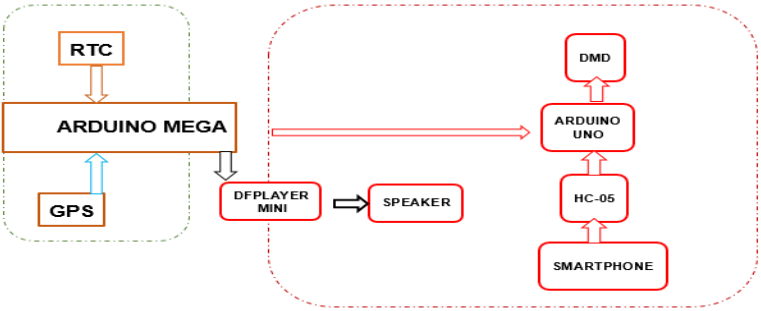


Gambar 2.7. Modul *DFPalyer Mini*

BAB III
PERANCANGAN DAN PEMBUATAN SISTEM

Pada Bab ini dibahas mengenai perancangan sistem *master clock* dengan sinkronisasi GPS dan komunikasi antara *master clock* dengan *slave clokc* serta penambahan fitur suara untuk membunyikan waktu tiap jam dan aplikasi *timer* pada *slave clock* yang dikendalikan menggunakan *smartphone*. Pada poin 3.2.2, 3.3.2 ,3.3.3, 3.3.4, 3.3.5, 3.3.6. merupakan tugas penelitian Candra Mashuri dan poin 3.2.3, 3.2.4, 3.2.5 dan 3.3.7, 3.3.8, 3.3.9, 3.3.10 merupakan tugas penelitian Samsul Hidayatulloh.

3.1 Blok Fungsional Sistem



Candra Mashuri
Samsul Hidayatulloh

Gambar 3.1. Blok Fungsional Sistem

Pada Gambar 3.1 agar sistem dapat berjalan dan berkomunikasi secara optimal, dibutuhkan beberapa komponen seperti dibawah ini:

1. GPS, berfungsi untuk sinkronisasi data waktu pada *master clock* dengan waktu pada GPS.
2. RTC DS 3231, berfungsi sebagai untuk menghitung waktu dengan *set* waktu menggunakan data dari GPS.
3. RS485, berfungsi sebagai komunikasi *serial* untuk mengirimkan data waktu dari *masterclock* menuju *slaveclock.*
4. DOT *Matrix* 16x64, berfungsi untuk menampilkan waktu pada *slaveclock.*
5. Arduino Uno, mikrokontroler yang berfungsi sebagai pusat pengendali, pengolah dan pemroses data waktu pada *slaveclock*.

6. Arduino Mega, mikrokontroler yang berfungsi sebagai pusat pengirim data waktu yang diambil pada GPS untuk dikirimkan kepada *slave clock*.
7. Modul *DFPlayer Mini* , merupakan modul yang mengolah data dari *micro* SD berupa format MP3 yang digunakan untuk menunjukkan suara tiap jam.
8. Modul HC-05 *bluetooth*, berfungsi untuk komunikasi antara *bluetooth* ke *slave clcock* ataupun sebaliknya, media komunikasi ini menggunakan media udara. Dengan adanya modul ini dimungkinkan untuk set *timer* pada *slave clock*.
9. *Speaker*, berfungsi sebagai *output* suara yang menunjukkan waktu tiap jam.

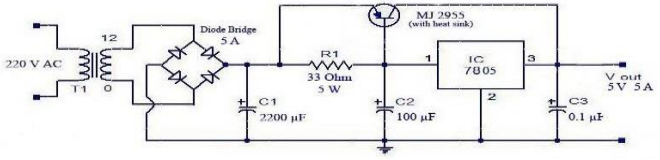
3.2 Perancangan Perangkat Keras

Dalam pembuatan perangkat keras ada beberapa komponen yang harus terhubung agar sistem *master clock* dapat berjalan dengan baik. Tahapan pembuatan tersebut adalah sebagai berikut:

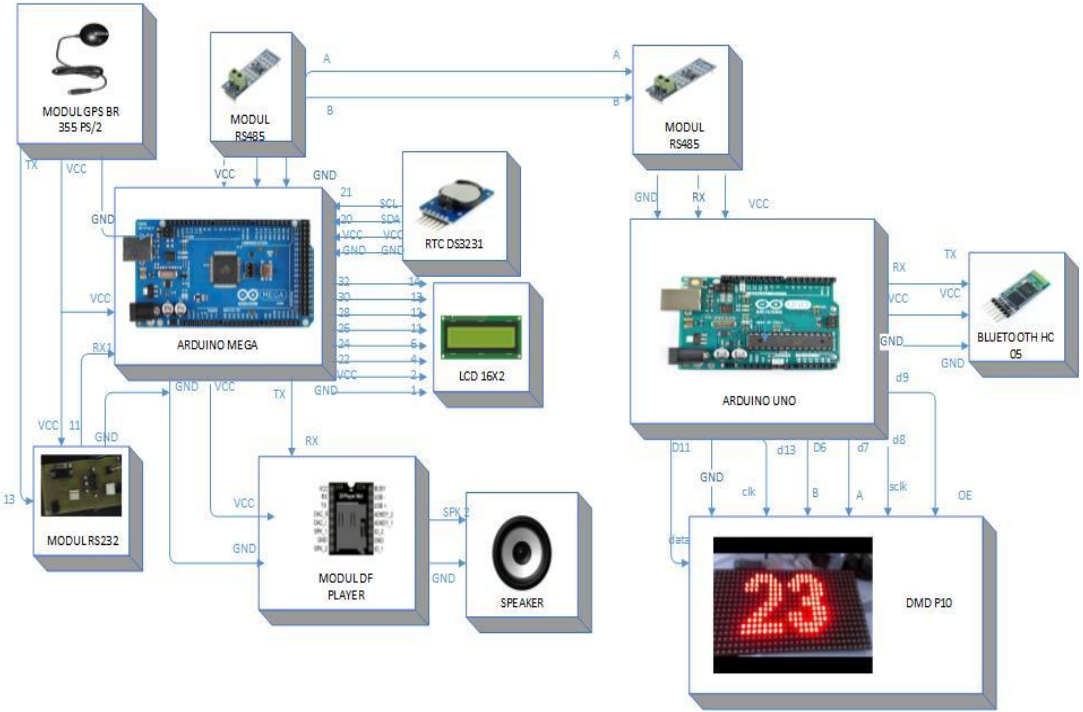
3.2.1 Skematik Sistem Elektronika

Pusat kendali dari *master clock* menggunakan mikrokontroler Arduino Mega yang mengambil data dari GPS digunakan untuk *set* waktu pada RTC tiap hari.

Pada *slave clock* menggunakan mikrokontroler Arduino Uno , data dari *master clock* akan dikirim menggunakan komunikasi *serial* RS 485 dan nanti akan dibaca dan diproses oleh Arduino Uno yang kemudian akan ditampilkan pada DOT *Matrix* 16x64. Fungsi *smartphone* digunakan untuk mengatur waktu *timer* yang nantinya diterima oleh modul *bluetooth* HC 05. *Timer* nanti akan diolah dan dijalankan oleh Arduino Uno dan ditampilkan pada DOT *Matrix* 16x64. Untuk *power supply* dari *slave clock* menggunakan tegangan dengan besar 5V dengan arus beban sampai 5A. Rangkaian *power supply* terlihat pada Gambar 3.2.



Gambar 3.2. Rangkaian *Power Supply*



Gambar 3.3. Skematik Diagram *Slave Clock*

Pada Gambar 3.3 Arduino Mega sebagai mikrokontroler pada *master clock* mempunyai 4 pin komunikasi *serial* sehingga dapat mengendalikan atau menerima data dari komponen yang menggunakan komunikasi *serial.* Pada *master clock* mengendalikan dan menerima data dari 3 alat yang menggunakan komunikasi *serial* yaitu GPS, RS 485 dan *DFplayer mini*. GPS menggunakan pin Rx1 pada Arduino Mega karena GPS hanya mengirim data ke Arduino Mega. Selain terhubung dengan pin Rx Arduino Mega, GPS juga mengambil sumber tegangan 5V dan *ground* dari Arduino Mega. Untuk RS 485 menggunakan pin Tx Arduino Mega untuk mengirim data ke *slave clock*. RS 485 juga mengambil sumber tegangan 5V pada Arduino Mega untuk mensuplai tegangan dan untuk mengirim data ke *slave clock.* Untuk *DFpalyer mini* menggunakan pin Tx3 Arduino Mega karena *file* suara pada *DFPlayer mini* akan dipanggil melalui perintah Arduino Mega dan juga terhubung pada pin 5V dan *ground* untuk mensuplai tegangan. Pada Arduino Mega juga terdapat pin I2C yaitu pin SDA dan SCL yang terhubung pada RTC DS 3231. Pada Arduino Uno sebagai mikrokontroler *slave clock* hanya terdapat satu komunikasi seial pada pin sehingga menggunakan *library* untuk membuat satu komunikasi *serial.* Pin komunikasi *serial* digunakan oleh dua komponen yaitu HC 05 *bluetooth* modul dan RS 485.

Untuk lebih memperjelas tentang koneksi antar pin komponen pada Aduino Uno dapat dilihat pada Tabel 3.1.

Tabel 3.1. Data *Mapping* I/O Arduino Mega dan Arduino Uno

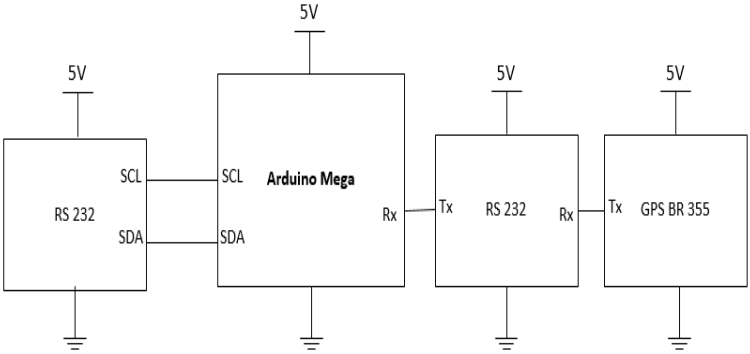
No.	**Komponen**	**Koneksi Pin**		
		Modul	**Arduino Mega**	**Arduino Uno**
1.	RS 485 pada *master clock*	RO DE RE 5V GND	Tx2 5V 5V 5V GND	
2.	RS 485 pada *slave clock*	DI DE RE 5V GND		Rx GND GND 5V GND

No.	Komponen	Koneksi Pin		
		Modul	**Arduino Mega**	**Arduino Uno**
3.	Modul *bluetooth* HC 05	3,3V GND Rx		3,3V GND A5
4.	Modul *DFPlayer mini*	5V Gnd Rx	5V Gnd Tx2	
5.	GPS BR 355	5V GND Tx	5V GND Rx1	
6.	DOT *Matrix Display* 16x64	5V GND A B OE CLK SCLK DATA		5V GND D6 D7 D9 D13 D8 D11
7.	RTC DS 3231	5V GND SDA SCL	5V GND D20 D21	
8	LCD 16x2	1 2 4 6 11 12 13 14	GND 5V D22 D24 D26 D28 D30 D32	

3.2.2 GPS BR 355 dan RTC DS 3231

GPS BR 355 membutuhkan *converter* RS 232 untuk terhubung dengan Arduino Mega. Pin yang digunakan GPS hanya pin VCC, GND, dan Tx. Pin Rx tidak dibutuhkan karena GPS hanya mengirim data ke

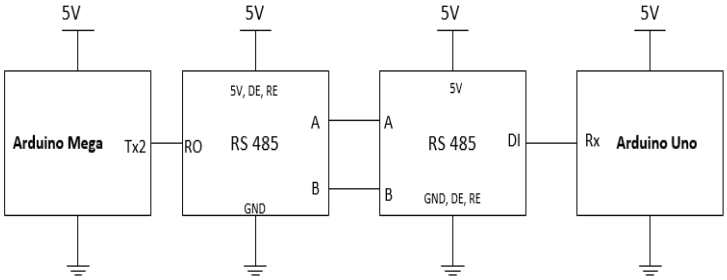
Arduino Mega. Lalu RTC DS 3231 terhubung dengan pin SDA dan SCL Arduino Mega. Skematik koneksi pin antara GPS, RTC DS 3231 dan Arduino Mega seperti pada Gambar 3.4.



Gambar 3.4. Skematik Rangkaian GPS BR 355

3.2.3 RS 485

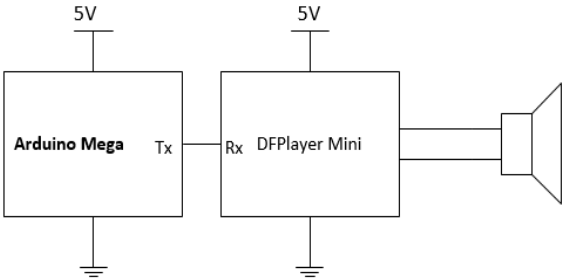
Komunikasi *serial* antara *master clock* dengan *slave clock* membutuhkan *converter* RS 485 agar komunikasi dapat berjalan pada jarak kurang dari 1200 m. Pin DE dan RE pada RS 485 yang terhubung dengan *master clock* harus diberi tegangan 5V karena mengirim data. Untuk pin DE dan RE pada RS 485 yang terhubung *slave clock* harus dihubungkan pada *ground* karena menerima data dari *master clock*. Untuk RS 485 *master clock* pin yang digunakan mengirim adalah pin RO dan untuk RS 485 *slave clock* pin yang digunakan adalah pin DI.



Gambar 3.5. Skematik Rangkaian RS 485

3.2.4 *DFPlayer Mini*

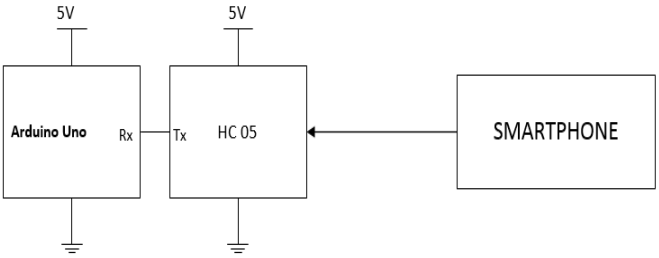
DFPlayer mini adalah modul MP3 yang digunakan untuk mengolah *file* .mp3 yang ada pada SD *card.* Pada proyek tugas akhir ini *DFPlayer mini* digunakan untuk membunyikan waktu tiap jam pada *master clock*. Pin yang digunakan pada Arduino Mega adalah pin Tx3 untuk mengirim perintah menjalankan suara sesuai *file. Output* dari *DFPlayer Mini* akan dikeluarkan menggunakan *speaker*. Skematik rangkaian terlihat pada Gambar 3.6.



Gambar 3.6. Skematik Rangkaian *DFPlayer Mini*

3.2.5 *Bluetooth* HC 05

Bluetooth HC 05 adalah modul *Bluetooth* yang bisa digunakan sebagai *slave* dan *master*. Pada proyek tugas akhir ini digunakan untuk *slave* karena hanya menerima perintah dari *smartphone*. Pin Tx akan terhubung pada Rx Arduino Uno karena HC 05 menerima data dari *smartphone* lalu mengirim data tersebut ke Arduino Uno. Skematik rangkaian HC 05 terlihat pada Gambar 3.7.



Gambar 3.7. Skematik Rangkaian HC 05

3.3 Pembuatan Perangkat Lunak

Dalam pembuatan perangkat lunak ada beberapa program yang harus dibuat agar sistem *master clock* dapat berjalan dengan baik. Tahapan pembuatan tersebut adalah sebagai berikut:

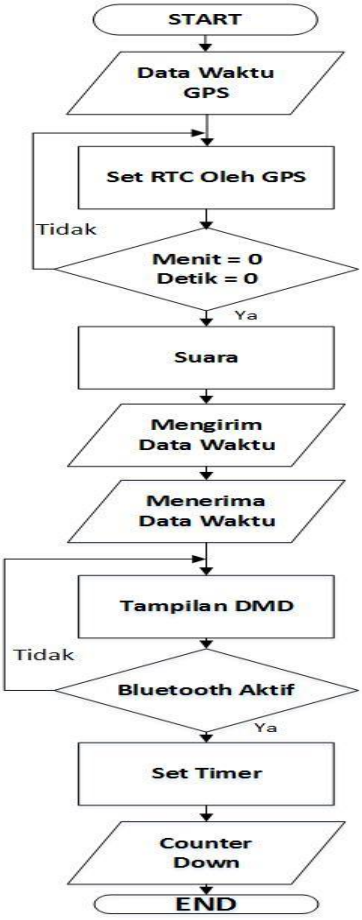
3.3.1 Pembuatan *Flowchart* Program

Flowchart merupakan bagan yang memperlihatkan urutan dan hubungan antar proses beserta instruksinya. Bagan ini dinyatakan dengan simbol. Dengan demikian setiap simbol menggambarkan proses tertentu. Sedangkan hubungan antar proses digambarkan dengan garis penghubung. *Flowchart* merupakan langkah awal pembuatan program. Dengan adanya *flowchart* urutan poses kegiatan menjadi lebih jelas.

Untuk pengolahan data dengan komputer, dapat dirangkum urutan dasar untuk pemecahan suatu masalah, yaitu;

- *START*: Berisi instruksi untuk persiapan perlatan yang diperlukan sebelum menangani pemecahan masalah.
- *READ*: Berisi instruksi untuk membaca data dari suatu peralatan.
- *PROCESS*: Berisi kegiatan yang berkaitan dengan pemecahan persoalan sesuai dengan data yang dibaca.
- *WRITE*: Berisi instruksi untuk merekam hasil kegiatan ke perlatan *output*.
- *END*: Mengakhiri kegiatan pengolahan.

Flowchart berupa sistem kesuluruhan. Pada bagian *parsing* data GPS dan *reset* pada RTC akan dilakukan oleh Candra Mashuri. Untuk pengiriman data dan penambahan fitur *master clock* akan dilakukan oleh Samsul Hidayatulloh. *Flowchart* akan seperti pada Gambar 3.8.



Gambar 3.8. *Flowchart* Sistem *Master Clock*

3.3.2 Program Sinkronisasi GPS

GPS akan mengirim data ke *master clock*, data yang ditangkap oleh *master clock* akan dipecah dan diambil data waktu. Data $GPRMC dan $GPGGA terdapat data waktu sehingga data tersebut akan dipecah.

```
String DataGps = "";
String GGA = ""; // string DataGps untuk menampung data DataGps dari GPS
String GPRMC = ""; // string DataGps untuk menampung data DataGps dari GPS

// parameter gps
String TimeGps;
String Latitude;
String DirLatitudeNorS;
String Longitude;
String DirLongitudeEorW;
String GpsQuality;

String GPRMC_Time;
String GPRMC_Date;
```

Gambar 3.9. Program Variabel untuk *Parsing* GPS

Pada Gambar 3.9 diuntunjukkan dengan membuat karakter untuk *parsing* data dari GPS. Pada proyek tugas akhir ini hanya mengambil data waktu maka variabel yang digunakan adalah *TimeGps. TimeGps* adalah data jam, menit dan detik dari $GPGGA dan $GPRMC.

```
/*
  Mengambil data GPS dari serial 1 arduino mega
*/
while (Serial1.available() > 0) {
  DataGps = Serial1.readStringUntil('\r');
}

/*
   Mengambil data $GPDataGps saja
*/
if (DataGps.substring(0, 7) == "\n$GPGGA") {
  //Serial.println(DataGps);
```

Gambar 3.10. Program Menangkap Data GPS

Pada Gambar 3.10. saat GPS aktif maka Arduino Mega akan menangkap data $GPGGA untuk kemudian *parsing* data waktu untuk *reset* RTC.

```
    if (debug) {
      Serial.print("\n======== GGA Data Gps ========\n");
      Serial.print("Time GPS\t:");
      Serial.println(TimeGps);

      Serial.print("Latitude\t:");
      Serial.print(Latitude);
      Serial.print(" - ");
      Serial.println(DirLatitudeNorS);
      Serial.print("Longitude\t:");
      Serial.print(Longitude);
      Serial.print(" - ");
      Serial.println(DirLongitudeEorW);
      Serial.print("GPS Quality\t:");
      if (GpsQuality == "0") Serial.print("fix not available");
      if (GpsQuality == "1") Serial.print("GPS fix");
      if (GpsQuality == "2") Serial.print("Differential GPS fix");
      Serial.print("\n\n");
    }
  }

/*
  data GPRMC
*/
if (DataGps.substring(0, 7) == "\n$GPRMC") {
  GPRMC = DataGps;

  /*
  if (debug) {
    if (GpsQuality == "1" || GpsQuality == "2") {
      Serial.print("\n======== GPRMC Data Gps ========\n");
      Serial.print("Date\t:");
      Serial.print(tanggal);
      Serial.print("/");
      Serial.print(bulan);
      Serial.print("/");
      Serial.println(tahun);

      Serial.print("Time\t:");
      Serial.print(jam);
      Serial.print(":");
      Serial.print(menit);
      Serial.print(":");
      Serial.println(detik);

    }
```

Gambar 3.11. Program *Parsing* GPS

Pada Gambar 3.11 data yang akan dipecah yaitu data waktu. Setelah menerima data waktu akan dirubah menjadi variabel yang nantinya untuk *reset* RTC.

3.3.3 Program *Set* RTC DS 3231

Pada Gambar 3.12 RTC akan di *set* sesuai dengan jam dan tanggal secara manual. Karena GPS belum terhubung. Setelah alat dinyalakan program akan diganti menjadi *reset* untuk RTC berdasarkan data GPS. Program seperti pada Gambar 3.12.

```
rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));

if (rtc.lostPower()) {
  Serial.println("RTC lost power, lets set the time!");
  // following line sets the RTC to the date & time this sketch was compiled
  rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  // This line sets the RTC with an explicit date & time, for example to set
  //January 21, 2014 at 3am you would call:
  // rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
}
```

Gambar 3.12. Program *Set* RTC DS 3231

3.3.4 Program Sinkronisasi GPS untuk *Set* RTC

Pada Gambar 3.13 RTC akan *reset* setiap jam 23.00 WIB oleh GPS sekaligus sinkronisasi RTC dengan GPS. Data waktu dari GPS yang akan *reset* waktu pada RTC. Terlihat seperti pada Gambar 3.13.

```
/*
   jadwal_update_waktu dalam satuan jam 0 - 23
*/
int jadwal_update_waktu = 23; // 23 pada saat jam 11  tengah malam

// nunggu gps fix
if (GpsQuality == "1" || GpsQuality == "2") {
  // update
  rtc.adjust(DateTime(tahun, bulan, tanggal, local_time, menit, detik)); // adjust untuk sinkron waktu dari gps ke rtc
  Serial.println("++++++++++++++++++++++++++++++\nwaktu telah diperbaharui dg gps!\n++++++++++++++++++++++++++++++");
  timeUpdateGpsFlag = 1;
 }
}
if (jam_rtc != jadwal_update_waktu) {
 timeUpdateGpsFlag = 0;
}
```

Gambar 3.13. Program GPS *Set* RTC Setiap Jam 23.00 WIB

3.3.5 Program *GMT* (*Greenwich Meridian Time*) + 7

Pada Gambar 3.14 data GPS waktu masih GMT+0 maka GMT akan di tambah 7 sesuai dengan Wilayah Indonesia Bagian Barat (WIB). *int UTC_TimeZone=7;* adalah zona waktu di Indonesia. Program set untuk GMT+7 seperti Gambar 3.14.

```
int UTC_TimeZone = 7;
```

Gambar 3.14. Program GMT+7

3.3.6 Program Menampilkan di LCD 16x2

Program akan menampilkan hasil jam dari pemrograman ke LCD 16X2. Jam yang ditampilkan ialah jam RTC dan ditampilkan GPS *fix* dan GPS *No fix* ari hasil dari pemecahan data $GPGGA yang nantinya akan diketahui kualitas sinyal GPS yang masuk apakah cepat *fix* (sinyal baik) atau tidak. Program terlihat pada Gambar 3.15.

```
// print ke lcd
String buff_jam = tampil_jam[jam_rtc] + ":" + tampil_jam[menit_rtc] + ":" + tampil_jam[detik_rtc];

if (timerStartTampil + 3000 > millis() ) {
  lcd.setCursor(0, 0);
  lcd.print ("Master Time&Date");
} else if (timerStartTampil + 6000 > millis()) {
  if (GpsQuality == "1" || GpsQuality == "2") {
    lcd.setCursor(0, 0);
    lcd.print("    GPS Fix!     ");
  }
  if (GpsQuality == "0") {
    lcd.setCursor(0, 0);
    lcd.print("   GPS no Fix!   ");
  }
  //timerStartTampil = millis();
} else if (timerStartTampil + 9000 > millis()) {
  timerStartTampil = millis();
}

lcd.setCursor(4, 1);
lcd.print(buff_jam);

// kirim ke slave
Serial2.print(buff_jam);
Serial2.print("\n");
```

Gambar 3.15. Program Menampilkan Di LCD 16x2

3.3.7 Program Pengiriman Data dari *Master Clock*

Pada Gambar 3.16 bagian ini akan membuat *software* untuk mengirim data waktu dari *master clock* ke *slave clock*. Pada komunikasi serial RS485 pin DE dan RE harus terhubung pada vcc agar bisa mengirim data waktu. *Baud rate* disetting 9600 agar sinkron dengan arduino. Pin TX dan RX RS485 terhubung pada TX2 dan RX2 arduino mega sehingga *setup* untuk pin menjadi Serial2.begin().

```
// kirim ke slave
Serial2.print(buff_jam);
Serial2.print("\n");
```

Gambar 3.16. Program untuk Mengirimkan Data ke *Slave Clock*

Untuk mengirimkan data waktu maka data yang diparsing dari *GPS* diubah dulu agar bisa dikirim. Lalu data dikirim menggunakan format Serial.print().

3.3.8 Program Suara pada *DFPlayer Mini*

Setelah mendapat data waktu dan menampilkannya ke LCD langkah selanjutnya ialah membaca data suara untuk menunjukkan suara tiap jam.

```
Serial3.begin(9600);
if (!myDFPlayer.begin(Serial3)) {
 while(true);
}
myDFPlayer.volume(25);  //Set volume value. From 0 to 30
```

Gambar 3.17. *Setup* untuk Modul *DFPlayer Mini*

Pada *setting default* modul *DFPlayer mini* suara akan otomatis berjalan ketika dikasih sumber tegangan dan *mode* yang digunakan adalah *continuous mode* sehingga program suara akan otomatis berganti setiap selesai. Maka dari itu, diperlukan *setup* yang digunakan untuk menghentikan suara agar setiap kali dikasih sumber tidak berbunyi dan untuk membuat modul menjadi *single mode*, program *setup* seperti Gambar 3.17. untuk *volume* bisa diatur dari 0 sampai 30 . Dan untuk *single mode* digunakan format program Serial.print("g\r"); agar program suara tidak otomatis ganti ke *file* berikutnya. Setelah *setup* program akan dipanggil sesuai jam saat itu. Program seperti pada Gambar 3.18.

```
//suara jam
if(buff_jam == "00:00:00") {myDFPlayer.play(12);}
if(buff_jam == "01:00:00") {myDFPlayer.play(1);}
if(buff_jam == "02:00:00") {myDFPlayer.play(2);}
if(buff_jam == "03:00:00") {myDFPlayer.play(3);}
if(buff_jam == "04:00:00") {myDFPlayer.play(4);}
if(buff_jam == "05:00:00") {myDFPlayer.play(5);}
if(buff_jam == "06:00:00") {myDFPlayer.play(6);}
if(buff_jam == "07:00:00") {myDFPlayer.play(7);}
if(buff_jam == "08:00:00") {myDFPlayer.play(8);}
if(buff_jam == "09:00:00") {myDFPlayer.play(9);}
if(buff_jam == "10:00:00") {myDFPlayer.play(10);}
if(buff_jam == "11:00:00") {myDFPlayer.play(11);}
if(buff_jam == "12:00:00") {myDFPlayer.play(12);}
if(buff_jam == "13:00:00") {myDFPlayer.play(1);}
if(buff_jam == "14:00:00") {myDFPlayer.play(2);}
if(buff_jam == "15:00:00") {myDFPlayer.play(3);}
if(buff_jam == "16:00:00") {myDFPlayer.play(4);}
if(buff_jam == "17:00:00") {myDFPlayer.play(5);}
if(buff_jam == "18:00:00") {myDFPlayer.play(6);}
if(buff_jam == "19:00:00") {myDFPlayer.play(7);}
if(buff_jam == "20:00:00") {myDFPlayer.play(8);}
if(buff_jam == "21:00:00") {myDFPlayer.play(9);}
if(buff_jam == "22:00:00") {myDFPlayer.play(10);}
if(buff_jam == "23:00:00") {myDFPlayer.play(11);}
```

Gambar 3.18. Perintah Pemanggilan *File* Suara

3.3.9 Pembuatan Program Komunikasi *Master Clock*

Pada bagian ini akan membuat *software* untuk menangkap data waktu dari *master clock* ke *slave clock*. Pada komunikasi *serial* RS485 pin DE dan RE harus terhubung pada *ground* agar bisa menerima data waktu untuk *slave clock*. Oleh karena itu disiasati dengan menggunakan pin 4 pada dengan mode *low* sehingga data dari *master clock* bisa masuk pada *slave clock.*

```
pinMode(4, OUTPUT);
digitalWrite(4, LOW); // mode low
dmd.setBrightness(20);
dmd.selectFont(ArialBold14);
dmd.begin();
```

Gambar 3.19. *Setup* Tampilan pada DMD

Untuk menampilkan *font* pada DMD menggunakan *library* DMD2 yang sudah ada pada arduino, *font* yang digunakan menggunakan *ArialBold14*. Dan untuk memulai dengan DMD.begin();

```
void Serial485() {
  if (Serial.available() > 0) {
    String dt = Serial.readStringUntil('\n');
    char dtf[25];
    dt.toCharArray(dtf, 25);
    dmd.clearScreen();
    dmd.drawString(5, 2, (dtf));
```

Gambar 3.20. Program untuk Menangkap dan Menampilkan Data

Pada Gambar 3.20 ketika data dari *master clock* masuk pada s*lave clock* maka *slave clock* akan membaca data tersebut. Data waktu akan ditampilkan pada DMD dengan format dmd.drawString (5, 2 (dtf)).

3.3.10 Pembuatan *Software* untuk *Timer*

Langkah selanjutnya ialah membuat program *timer* yang dikendalikan menggunakan *smartphone*. Komunikasi yang digunakan pada *smartphone* dengan android yaitu menggunakan *bluetooth*. *Bluetooth* yang digunakan pada arduino adalah modul HC 05. Untuk mengaktifkan modul *bluetooth* HC 05 dibutuhkan *setup* pada Arduino Uno seperti pada Gambar 3.21.

```
bt.begin(4800);
Serial.begin(9600);

timer_tick = millis();
```

Gambar 3.21. *Setup* HC 05 pada Arduino

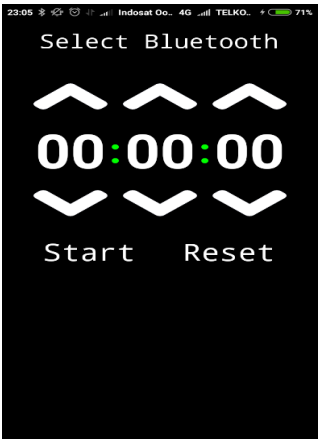
Pada *setup* HC 05 menggunakan pin tx dan rx yang disiasati menggunakan pin D5 dan D4 dengan menggunakan format *setup SoftwareSerial* (Tx,Rx) sesuai dengan pin yang digunakan.

```
if (bt.available() > 0) {
  timeout_bt = 10;
  String buff = bt.readStringUntil('0');

  Serial.println(buff);
  if (buff == "1") {
    cnt_jam++;
  }
  if (buff == "2") {
    cnt_jam--;
  }
  if (buff == "3") {
    cnt_menit++;
  }
  if (buff == "4") {
    cnt_menit--;
  }
  if (buff == "5") {
    cnt_detik++;
  }
  if (buff == "6") {
    cnt_detik--;
  }
  if (buff == "7") {
    flag_start = 1;
    start_millis = millis();
  }
  if (buff == "8") {
    flag_start = 0;
    cnt_jam = 0;
    cnt_menit = 0;
    cnt_detik = 0;
    dmd.clearScreen();
    dmd.drawString(5, 2, F("RESET!"));
    sDelay(3000);
  }

  if (cnt_jam < 0) cnt_jam = 24;
  if (cnt_jam > 24) cnt_jam = 0;
  if (cnt_menit < 0) cnt_menit = 59;
  if (cnt_menit > 59) cnt_menit = 0;
  if (cnt_detik < 0) cnt_detik = 59;
  if (cnt_detik > 59) cnt_detik = 0;
```

Gambar 3.22. *Set* Waktu *Timer* pada Arduino

Pada Gambar 3.22 ditunjukkan ketika *Bluetooth* aktif maka program *timer* akan berjalan. Perintah-perintah yang digunakan menyesuaikan dengan program yang ada pada *smartphone* seperti perintah *reset* , *smartphone* akan mengirim karakter 8 untuk peritah *reset* maka Arduino Uno harus menyesuiakan dengan karakter 8 agar perintah *reset* aktif. Untuk *set* waktu *timer* maka digunakan karakter 1-6, untuk menjalankan program *counter timer* maka diperlukan karakter 7 dan untuk perintah *reset* digunakan karakter 8.

Software mengambil pada aplikasi *CDTimer* seperti pada Gambar 3.23. untuk mengatur waktu *timer* dari jam, menit dan detik.



Gambar 3.23. Aplikasi *CDTimer* pada *Smartphone*

Ketika kita pilih *select bluetooth*, modul *bluetooth* HC 05 akan aktif sebagai *slave* dan mengaktifkan program *set timer.*

```
if (cnt_jam == 0 && cnt_menit == 0 && cnt_detik == 0) {
  flag_start = 0;
  timeout_bt = 0;
  for (int i = 0; i < 3; i++) {
    bt.println("waktu habis!");
    dmd.clearScreen();
    sDelay(500);
    dmd.drawString(5, 2, F("WAKTU"));
    sDelay(2000);
    dmd.clearScreen();
    sDelay(100);
    dmd.drawString(5, 2, F("HABIS!"));
    sDelay(3000);
  }
  dmd.clearScreen();
}
```

Gambar 3.24. Program *Counter Down* Selesai

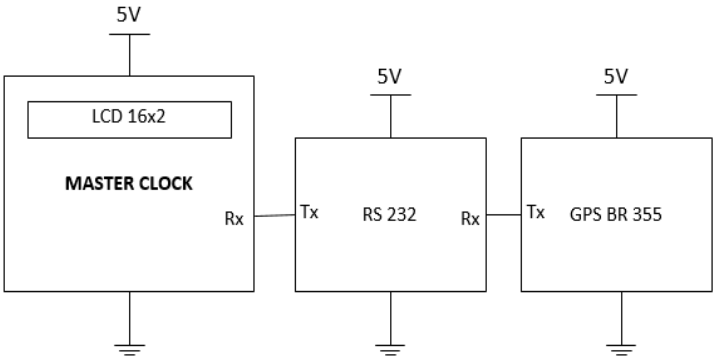
Pada Gambar 3.24 saat tombol *start* ditekan maka *counter down* akan aktif dan ketika *counter down* selesai maka akan muncul tampilan "waktu habis". Ketika 10 detik tidak ada perintah dari *smartphone* maka program akan otomatis kembali ke program waktu.

BAB IV
HASIL IMPLEMENTASI ALAT

Bab ini akan dijelaskan mengenai uji coba pada masing masing komponen dan juga data dari komponen. Pada bab ini Candra Mashuri bertangung jawab pada pegujian yang berhubungan dengan sinkronisasi GPS maupun RTC terdapat pada poin 4.1 dan 4.2. Begitu juga dengan Samsul Hidayatulloh bertanggung jawab pada pengujian yang berhubungan dengan komunikasi antara *master clock* dengan *slave clock*, serta penambahan fitur pada *master clock* terdapat pada poin 4.3, 4.4, 4.5. Tiap poin dijelaskan siapa yang bertanggung jawab.

4.1 Pengujian Akses Data GPS

Cara pengujian akses data GPS ialah untuk mengetahui kecepatan sinkronisasi *master clock* degan GPS saat pertama kali dinyalakan, adalah dengan melihat tampilan LCD yang berada di *Master Clock* . Pengujian dilakukan di 4 kondisi tempat yang berbeda-beda. Skematik untuk rangakian pengujian akses data GPS ada pad Gambar 4.1.



Gambar 4.1. *Wiring* Pengujian Akses Data GPS

Pada saat pertama kali dinyalakan akan muncul tulisan ”GPS *no Fix*!” pada LCD. Gambar 4.2. menunjukkan tampilan pengujian saat baru dinyalakan dan belum sinkronisasi dengan GPS. Program untuk sinkronisasi GPS akan aktif ketika GPS mulai mengirimkan data ke *master clock*. Program seperti pada Gambar 3.11. Setelah GPS mulai sinkronisasi dengnan master clock akan terlihat tulisan “GPS *Fix!”* pada

LCD seperti Gambar 4.3. Selama rentang waktu dari saat pertama kali dinyalakan sampai GPS mulai sinkronisasi akan dicatat . Kondisi alat menunjukkan bahwa apakah GPS akan selalu sinkronisasi. Kondisi alat baik menunjukkan bahwa GPS selalu sinkronisasi dengan *master clock* sampai *master clock* dimatikan. Kondisi alat buruk ketika GPS dan *master clock* mengalami gangguan pada sinkronisasi.

Gambar 4.2. Tampilan *Master Clock* Ketika Kondisi GPS *No Fix!*

Gambar 4.3. Tampilan *Master Clock* Ketika Kondisi GPS *Fix!*

Tabel 4.1. Pengujian Dalam Ruangan

No.	Waktu GPS *No Fix*	Waktu GPS *Fix*	Rentang Waktu	Kondisi Alat
1.	11:13:20	11:36:23	23 Menit 3 Detik	Baik
2.	19:49:35	20:53:04	1 Jam 3 Menit 29 Detik	Baik
3.	21:56:13	22:00:30	3 Menit 17 Detik	Baik
4.	22:26:37	22:32:46	6 Menit 9 Detik	Baik
5.	22:34:20	22:41:49	7 Menit 29 Detik	Baik
6.	11:44:00	11:47:58	3 Menit 58 Detik	Baik
7.	11:53:08	11:56:34	3 Menit 29 Detik	Baik
8.	12:07:08	12:15:32	8 Menit 24 Detik	Baik
9.	12:30:05	12:33:02	2 Menit 57 Detik	Baik

No.	Waktu GPS *No Fix*	Waktu GPS *Fix*	Rentang Waktu	Kondisi Alat
10.	12:46:57	12:49:45	2 Menit 48 Detik	Baik
11.	11:38:37	11:41:48	3 Menit 11 Detik	Baik
12.	11:50:14	11:54:04	3 Menit 50 Detik	Baik
13.	12:22:37	12:29:20	6 Menit 43 Detik	Baik
14.	12:36:16	12:39:06	2 Menit 50 Detik	Baik
15.	12:51:44	12:57:24	5 Menit 40 Detik	Baik

Pada Tabel 4.1 menunjukkan rentang waktu ketika *master clock* mulai dinyalakan sampai tersinkronisasi dengan GPS membutuhkan waktu paling lama yaitu 1 jam 3 menit 29 detik dan paling cepat yaitu 2 menit 50 detik. Kondisi alat menunjukkan baik karena *master clock* selalu tersinkronisasi dengan GPS tanpa ada gangguan.

Tabel 4.2. Pengujian Luar Ruangan Diantara Gedung Bertingkat

No.	Waktu GPS *No Fix*	Waktu GPS *Fix*	Rentang Waktu	Kondisi Alat
1.	13:40:50	13:41:32	42 Detik	Baik
2.	13:46:29	13:47:42	1 Menit 13 Detik	Baik
3.	13:54:22	13:54:58	36 Detik	Baik
4.	13:58:42	13:59:04	22 Detik	Baik
5.	14:03:27	14:04:33	1 Menit 6 Detik	Baik
6.	13:43:11	13:44:14	1 Menit 3 Detik	Baik
7.	13:50:10	13:52:02	1 Menit 52 Detik	Baik
8.	13:55:41	13:56:53	1 Menit 12 Detik	Baik
9.	14:00:42	14:01:00	18 Detik	Baik
10.	14:05:30	14:06:20	50 Detik	Baik
11.	13:45:15	13:45:51	36 Detik	Baik

No.	Waktu GPS *No Fix*	Waktu GPS *Fix*	Rentang Waktu	Kondisi Alat
12.	13:52:40	13:53:38	58 Detik	Baik
13.	13:57:24	13:57:50	26 Detik	Baik
14.	14:02:10	14:02:50	40 Detik	Baik
15.	14:13:12	14:14:18	1 Menit 6 Detik	Baik

Pada Tabel 4.2 menunjukkan rentang waktu ketika *master clock* mulai dinyalakan sampai tersinkronisasi dengan GPS membutuhkan waktu paling lama yaitu 1 menit 52 detik dan paling cepat yaitu 18 detik. Kondisi alat menunjukkan baik karena *master clock* selalu tersinkronisasi dengan GPS tanpa ada gangguan.

Tabel 4.3. Pengujian Dalam Gedung Bertingkat

No.	Waktu GPS *No Fix*	Waktu GPS *Fix*	Rentang Waktu	Kondisi Alat
1.	16:01:05	16:02:16	1 Menit 11 Detik	Baik
2.	16:04:43	16:05:20	37 Detik	Baik
3.	21:33:26	21:33:54	28 Detik	Baik
4.	21:40:55	21:41:45	50 Detik	Baik
5.	21:42:57	21:44:14	1 Menit 17 Detik	Baik
6.	16:06:20	16:09:11	2 Menit 51 Detik	Baik
7.	21:14:38	21:15:28	50 Detik	Baik
8.	21:22:07	21:23:54	1 Menit 47 Detik	Baik
9.	21:24:45	21:25:28	43 Detik	Baik
10.	21:26:52	21:27:24	32 Detik	Baik
11.	16:10:47	16:11:42	55 Detik	Baik
12.	21:28:55	21:29:26	31 Detik	Baik
13.	21:30:32	21:31:03	31 Detik	Baik

No.	Waktu GPS *No Fix*	Waktu GPS *Fix*	Rentang Waktu	Kondisi Alat
14.	21:35:14	21:35:45	31 Detik	Baik
15.	21:37:23	21:38:50	1 Menit 27 Detik	Baik

Pada Tabel 4.3 menunjukkan rentang waktu ketika *master clock* mulai dinyalakan sampai tersinkronisasi dengan GPS membutuhkan waktu paling lama yaitu 2 menit 51 detik dan paling cepat yaitu 31 detik. Kondisi alat menunjukkan baik karena *master clock* selalu tersinkronisasi dengan GPS tanpa ada gangguan.

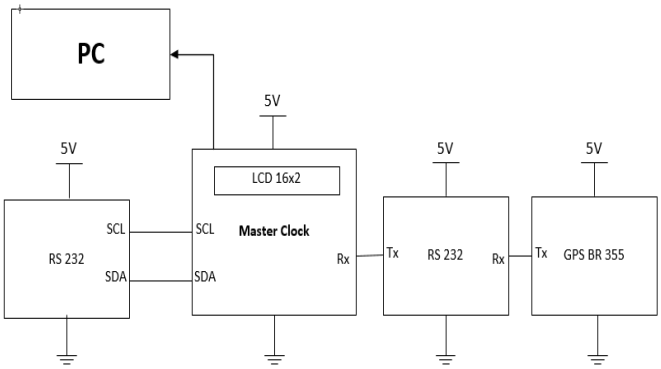
Tabel 4.4. Pengujian di Ruang Terbuka

No.	Waktu GPS *No Fix*	Waktu GPS *Fix*	Rentang Waktu	Kondisi Alat
1.	15:50:10	16:50:38	28 Detik	Baik
2.	15:55:14	15:55:40	26 Detik	Baik
3.	16:02:17	16:02:33	16 Detik	Baik
4.	16:04:38	16:04:59	21 Detik	Baik
5.	16:10:11	16:10:28	17 Detik	Baik
6.	08:30:14	08:30:45	31 Detik	Baik
7.	08:34:40	08:34:58	18 Detik	Baik
8.	08:35:08	08:35:34	26 Detik	Baik
9.	08:36:17	08:36:36	19 Detik	Baik
10.	08:37:20	08:37:35	16 Detik	Baik
11.	09:05:10	09:05:37	27 Detik	Baik
12.	09:07:24	09:07:48	24 Detik	Baik
13.	09:10:28	09:10:43	16 Detik	Baik
14.	09:13:05	09:13:21	16 Detik	Baik
15.	09:16:10	09:16:25	16 Detik	Baik

Pada Tabel 4.3 menunjukkan rentang waktu ketika *master clock* mulai dinyalakan sampai tersinkronisasi dengan GPS membutuhkan waktu paling lama yaitu 28 detik dan paling cepat yaitu 16 detik. Kondisi alat menunjukkan baik karena *master clock* selalu tersinkronisasi dengan GPS tanpa ada gangguan.

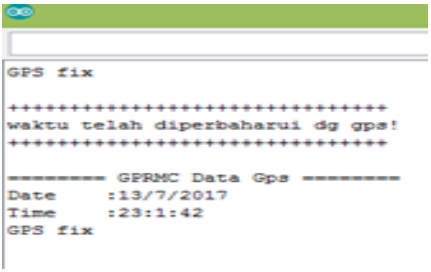
4.2 Pengujian GPS Sinkronisasi RTC DS 3231

Cara Pengujian dilakukan dengan mengamati waktu yang muncul di *Serial Monitor* Arduino Mega *(master clock*) hasil dari program GPS *set* RTC seperti pada Gambar 3.13. Tulisan Waktu telah diperbaharui dengan GPS adalah tanda bahwa RTC sudah *reset* atau disinkronisasi oleh GPS dengan asumsi GPS menunggu *Fix* (kualitas sinyal GPS) dan jika belum *fix* maka proses *reset* sedikit lama. Skematik Rangkaian Sinkronisasi RTC DS32321 terdapat pada Gambar 4.4.



Gambar 4.4. *Wiring* Pengujian GPS dengan RTC DS 3231

Pengujian dilakukan pada pukul 23.00 WIB dan dilakukan di 4 kondisi tempat yang berbeda. Dalam *serial monitor* muncul tulisan " waktu telah diperbaharui dg gps !" menandakan bahwa proses sinkronisasi telah berjalan dengan baik. Dalam rentang waktu antara waktu yang ada pada *master clock* dengan waktu yang ada pada PC akan dicatat sebagai perbandingan lama waktu *reset* RTC. Hasil *reset* RTC terlihat pada Gambar 4.5.



Gambar 4.5. Tampilan pada *Serial Monitor* Ketika *Reset* RTC

Tabel 4.5. Pengujian Sinkronisasi

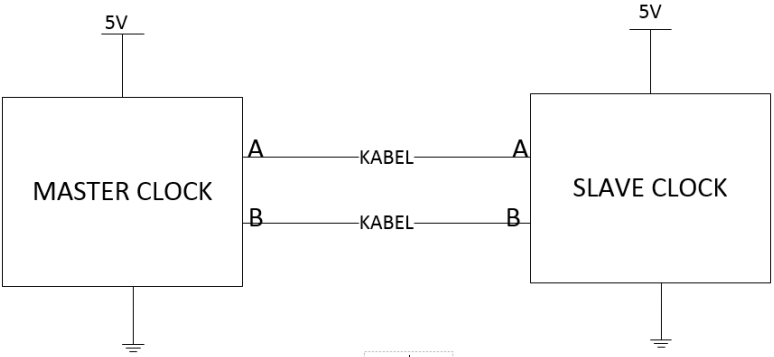
No.	Waktu GPS Sinkronisasi RTC	Waktu PC (Waktu Acuan Referensi)	Rentang Waktu	Tempat
1.	23:01:42	23:03:03	1 Menit 21 Detik	Dalam Ruangan
2.	23:30:05	23:33:02	2 Menit 57 Detik	Dalam Ruangan
3.	23:46:57	23:49:45	2 Menit 48 Detik	Dalam Ruangan
5.	23:38:37	23:41:48	3 Menit 11 Detik	Dalam Ruangan
6.	23:01:20	23:01:44	24 Detik	Luar Ruangan
7.	23:40:50	23:41:32	42 Detik	Luar Ruangan
8.	23:58:42	23:59:04	22 Detik	Luar Ruangan
9.	23:54:22	23:54:58	36 Detik	Luar Ruangan
10.	23:00:48	23:01:14	26 Detik	Gedung Bertingkat
11.	23:40:55	23:41:45	50 Detik	Gedung Bertingkat
12.	23:04:43	23:05:20	37 Detik	Gedung Bertingkat
13.	23:33:26	23:33:54	28 Detik	Gedung Bertingkat
14.	23:10:47	23:11:42	55 Detik	Gedung Bertingkat
15.	23:00:04	23:00:20	16 Detik	Ruang Terbuka

No.	Waktu GPS Sinkronisasi RTC	Waktu PC (Waktu Acuan Referensi)	Rentang Waktu	Tempat
16.	23:10:28	23:10:43	16 Detik	Ruang Terbuka
17.	23:13:05	23:13:21	16 Detik	Ruang Terbuka

Dari Tabel 4.5 dapat diketahui bahwa proses *reset* atau sinkronisasi RTC oleh GPS yang tercepat untuk *update* ialah yang berada di ruang terbuka kurang lebih 16 Detik.

4.3 Pengujian Pembacaan Akurasi Data Waktu

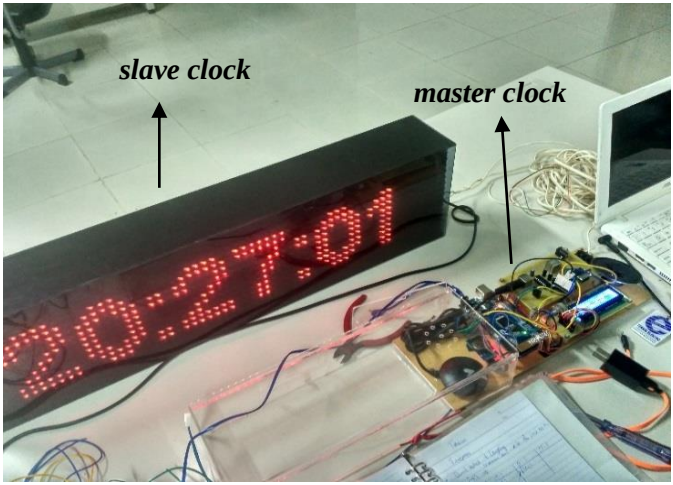
Pada pengujian ini dilakukan untuk mengetahui apakah *clock* pada *master clock* sama dengan *clock* pada *slave clock*. Cara melakukan pengujian dengan menghubungkan RS 485 pada *master clock* dengan RS 485 pada *slave clock.* Pin A pada *master clock* akan terhubung dengan Pin A pada *slave clock. Wiring* antara *master clock* dengan *slave clock* ada pada Gambar 4.6.



Gambar 4.6. *Wiring* Antara *Master Clock* dengan *Slave Clock*

Kabel yang digunakan selama pengujian mempunyai panjang berbeda mulai dar 3 m sampai 60 m. Setelah *master clock* dihubungkan dengan *slave clock,* lalu dinyalakan. *Master clock* akan mengirimkan dta le *slave clock.* Program untuk mengirimkan data waktu seperti pada Gambar 3.16. Waktu yang akan ditampilkan pada *master clock* dicatat dan dibandingkan dengan waktu yang ada pada *slave clock.* Kondisi alat

lancar ketika tidak ada *delay* waktu lebih dari 1 *second* pada *master clock* dengan *slave clock.* Kondisi alat tidak lancar ketika ada *delay* waktu lebih dari 1 *second* pada *master clock* dengan *slave clock.* Pengujian alat seperti pada Gambar 4.7.



Gambar 4.7. Pengujian Akurasi Data Waktu

Tabel 4.6. Hasil Komunikasi *Master Clock* dengan *Slave Clock*

Panjang kabel (m)	**Kondisi Pengiriman data waktu**	**Waktu pada *master clock***	**Waktu pada *slave Clock***
3	Lancar	07:02:03	07:02:03
		07:30:45	07:30:45
		07:45:02	07:45:02
10	Lancar	09:23:03	09:23:03
		09:38:48	09:38:48
		10:15:27	10:15:27

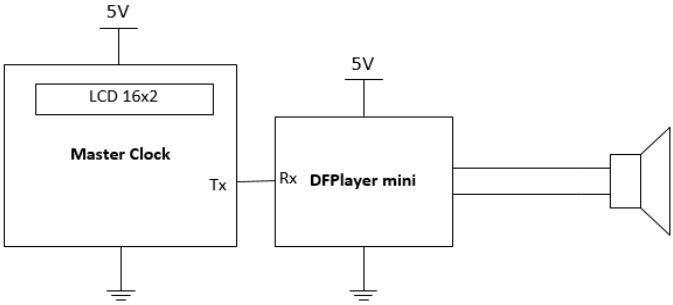
Panjang kabel (m)	Kondisi Pengiriman data waktu	Waktu pada *master clock*	Waktu pada *slave Clock*
20	Lancar	11:25:13	19:58:21
		11:27:23	20:03:01
		11:30:45	20:10:17
30	Lancar	11:39:51	11:39:51
		11:45:46	11:45:46
		11:54:19	11:54:19
40	Lancar	12;03:25	12;03:25
		12:07:39	12:07:39
		12:09:43	12:09:43
50	Lancar	12:15:59	12:15:59
		12:18:68	12:18:68
		12:23:47	12:23:47
60	Lancar	12:29:24	12:29:24
		12:37:17	12:37:17
		12:56:35	12:56:35
		13:00:03	13:00:03

Dari Tabel 4.6 hasil pengukuran pengiriman data dari *master clock* ke *slave clock* ada 6 percobaan menggunakan kabel yang mempunyai panjang berbeda. Pada kabel 3 m dengan mengambil 3 data dari waktu yang berbeda, waktu pada *master clock* dengan waktu yang ada pada *slave clock* masih menunjukkan waktu yang sama. Pada percobaan kedua menggunakan kabel 10 m masih tetap sama. Begitu juga dengan percobaan selanjutnya sampai percobaan terakhir dengan panjang kabel 60 m waktu yang ada pada *master clock* dengan *slave clock* masih sama sehingga tidak ada perbedaan waktu pada *master clock* dengan *slave clock.* Kondisi alat lancar menunjukkan bahwa waktu yang ada pada

master clock dengan waktu yang ada pada *slave clock* mempunyai *delay* kurang dari 1 *second.*

4.4 Pengujian Suara pada *Master Clock*

Pada pengujian ini akan dilakukan mencoba suara tiap jam apakah suara yang ditunjukkan sesuai dengan format jam yang tersedia. Pengujian dilakukan dengan menghubungkan pin *serial* dari *master clock* ke *DFPlayer mini*, data waktu pada *master clock* akan dikirimkan dan diolah untuk mengaktifkan *DFPlayer mini. DFPlayer mini* akan mengaktifkan *file* pada SD *card* lalu *file* tersebut akan dikeluarkan melalui *speaker* yang terhubung pada *DFPlayer mini.* Skematik rangkaian pengujian suara terdapat pada Gambar 4.8. Program untuk menjalankan suara pada *DFPlayer mini* terdapat pada Gambar 3.18. Nama *file* yang akan berbunyi sesuai jam terlihat pada Tabel 4.7.



Gambar 4.8. *Wiring* Pengujian Suara

Tabel 4.7. Format *File* Suara Sesuai Waktu

No.	Waktu	*File* suara pada SD *Card* (.mp3)
1	01:00:00	0001
2	02:00:00	0002
3	03:00:00	0003
4	04:00:00	0004
5	05:00:00	0005
6	06:00:00	0006

No.	Waktu	*File* suara pada SD *Card* (.mp3)
7	07:00:00	0007
8	08:00:00	0008
9	09:00:00	0009
10	10:00:00	0010
11	11:00:00	0011
12	12:00:00	0012
13	13:00:00	0001
14	14:00:00	0002
15	15:00:00	0003
16	16:00:00	0004
17	17:00:00	0005
18	18:00:00	0006
19	19:00:00	0007
20	20:00:00	0008
21	21:00:00	0009
22	22:00:00	0010
23	23:00:00	0011
24	00:00:00	0012

Pengujian dilakukan setiap jam selama 24 jam untuk mengetahui suara dari tiap jam tersebut. Kondisi alat berhasil ketika *file* suara yang berbunyi sesuai dengan waktu yang diatur pada program. Dan kondisi alat gagal ketika *file* suara berbunyi tidak sesuai waktu yang telah diatur pada program. Hasil pengujian terdapat pada Tabel 4.8 .

Tabel 4.8. Hasil Pengujian Suara pada Tiap Jam

No.	Waktu	Kondisi	File suara (.mp3)
1	01:00:00	Berhasil	0001
2	02:00:00	Berhasil	0002
3	03:00:00	Berhasil	0003
4	04:00:00	Berhasil	0004

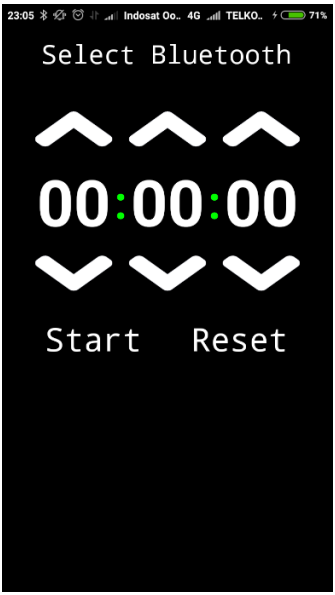
No.	Waktu	Kondisi	File suara (.mp3)
5	05:00:00	Berhasil	0005
6	06:00:00	Berhasil	0006
7	07:00:00	Berhasil	0007
8	08:00:00	Berhasil	0008
9	09:00:00	Berhasil	0009
10	10:00:00	Berhasil	0010
11	11:00:00	Berhasil	0011
12	12:00:00	Berhasil	0012
13	13:00:00	Berhasil	0001
14	14:00:00	Berhasil	0002
15	15:00:00	Berhasil	0003
16	16:00:00	Berhasil	0004
17	17:00:00	Berhasil	0005
18	18:00:00	Berhasil	0006
19	19:00:00	Berhasil	0007
20	20:00:00	Berhasil	0008
21	21:00:00	Berhasil	0009
22	22:00:00	Berhasil	0010
23	23:00:00	Berhasil	0011
24	00:00:00	Berhasil	0012

Pada pengujian suara tiap jam, dilakukan pengambilan data sesuai jam yang ditunjukkan. Kondisi alat dinyatakan berhasil karena *file* suara berjalan sesuai dengan waktu yang ditunjukkan.

4.5 Pengujian Aplikasi *Timer*

Pada pengujian ini untuk mengatur waktu *counter down* pada *slave clock* menggunakan *smartphone*. Koneksi antara *smartphone* dengan *slave clock* menggunakan *bluetooth.* Pada *slave clock* terdapat modul *bluetooth* HC 05 akan untuk komunikasi *wireless.* Modul *bluetooth* HC 05 akan menerima data dari *smartphone* menggunakan aplikasi *CDTimer*.

Program untuk menerima perintah dari *smartphone* seperti pada Gambar 3.22. Aplikasi *CDTimer* terlihat seperti Gambar 4.9.

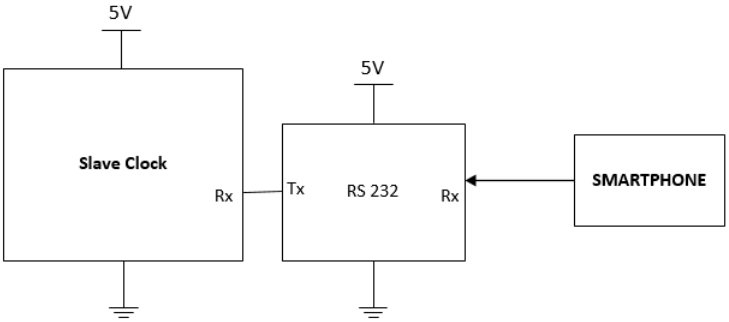


Gambar 4.9. Aplikasi *CDTimer*

Untuk mengatur waktu *counter down* pada *smartphone* dapat dilakukan dengan cara berikut:

1. Buka aplikasi *CDTimer* pada *smartphone*.
2. Aktifkan *bluetooth* pada *smartphone*, tekan tombol *select bluetooth* pada aplikasi *CD Timer*.
3. Pilih *device bluetooth* yang akan dihubungkan, pada Tugas akhir ini *device bluetooth* bernama HC 05.
4. Tekan tombol *reset* lalu tunggu sampai muncul kata *reset* pada *slave clock.*
5. *Set* waktu lalu tekan tombol *start*

Skematik rangkaian pengujian *timer* menggunakan *smartphone* terdapat pada Gambar 4.10. Pengujian *set* waktu menggunakan *smartphone* terlihat pada Gambar 4.11.



Gambar 4.10. *Wiring* Pengujian *Timer Slave Clock*

Gambar 4.11. Pengujian *Set Timer* Menggunakan *Smartphone*

Tabel 4.9. Hasil Pengujian *Set Timer* Menggunakan *Smartphone*

No	Waktu *set* pada *smartphone*	Waktu *set* pada *slave clock*	Kondisi
1	16:01:03	08:01:02	Gagal
2	07:07:08	03:04:04	Gagal
3	04:52:59	04:52:59	Gagal
4	05:06:02	02:05:02	Gagal
5	22:01:01	22:01:01	Berhasil
6	20:50:12	12:39:01	Gagal
7	06:05:04	06:04:02	Gagal

Pada Tabel 4.9. dengan melakukan pengujian selama 7 kali. Sebanyak 6 kali kondisi gagal karena waktu yang ada pada tampilan *smartphone* tidak sesuai dengan waktu yang ada tampilan *slave clock.* Dan sekali percobaan dengan kondisi berhasil karena waktu yang ada pada tampilan *smartphone* sesuai dengan waktu yang ada tampilan *slave clock.*

Selanjutnya yaitu pengujian *counter down* pada *slave clock.* Ketika waktu *timer* pada *slave clock* telah diatur, maka tombol *start* pada *smartphone* ditekan maka *counter down* pada *slave clock* akan berjalan. Program untuk menjalankan *counter down* terdapat pada Gambar 3.24. Kondisi berhasil ketika *counter down* pada *slave clock* selesai sesuai dengan waktu yang diatur menggunakan *smartphone.* Kondisi berhasil ketika *counter down* pada *slave clock* selesai tidak sesuai dengan waktu yang diatur menggunakan *smartphone.* Pengujian *counter down* terlihat pada Gambar 4.12. dan hasil pengujian terdapat pada Tabel 4.10.

Gambar 4.12. Pengujian *Counter Down* pada *Slave Clock*

Tabel 4.10. Hasil Pengujian *Counter Down*

No	*Counter Down pada Slave Clock*	Kondisi
1	00:00:32	Berhasil
2	00:15:00	Berhasil
3	00:30:00	Berhasil
4	01:30:00	Berhasil
5	02:00:00	Berhasil

Halaman ini sengaja dikosongkan

BAB V
KESIMPULAN

5.1 Kesimpulan

Dari hasil pengujian sistem *master clock* dapat diambil kesimpulan sebagai berikut:

1. Pada sinkronisasi GPS waktu terlama pada pengujian di dalam ruangan membutuhkan waktu lama untuk GPS terhubung ke satelit dengan waktu 1 jam 3 menit 29 detik. Dan untuk waktu tercepat pada tempat terbuka dengan waktu 16 detik. Sehingga RTC dibutuhkan agar penghitung waktu tetap berjalan walaupun GPS belum terhubung
2. Untuk komunikasi antara *master clock* dengan *slave clock* tidak ada perbedaan waktu ketika panjang kabel 60 m.
3. Suara pada tiap jam dapat berbunyi tepat waktu sesuai dengan jam tersebut.
4. Aplikasi *timer* pada *slave clock* dapat dikendalikan menggunakan *smartphone.*

5.2 Saran

Dari hasil perancangan tugas akhir ini masih kurang sempurna sehingga ada beberapa yang harus diperbaiki agar hasil tugas akhir ini medekati sempurna yaitu:

1. Menggunakan GPS yang waktu sinkronisasinya kurang dari 16 detik.
2. Komunikasi antara *master clock* dengan *slave clock* menggunakan *wireless.*

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] Wagner, Kenneth D. Clock System Design. **IEEE Design and Test of Computers**. IBM Corp. 1988.

[2] Hardianto, Derrie. Komunikasi Antar IC dengan Teknik I2C Bus Menggunakan PCF8574 pada Sistem Mikrokontroller AT89C2051. **Tugas Akhir**. Program Studi Teknik Elektro. Fakultas Teknik. Universitas Mercu Buana. Jakarta :2006.

[3] Simons, Barbara, Jennifer Lundelius Welch, Nancy Lynch. An Overview of Clock Synchronization. **Paperwork**. IBM Almaden Research Center. 1990.

[4] Nugroho, Ari. Rancang Bangun dan Pemrograman Sistem Transmisi Data GPS Menggunakan Teknologi CSD sebagai Aplikasi Sistem Penjejakan Posisi Berbasis Mikrokontroler AVR-ATMEGA8535. **Tugas Akhir.** Departemen Teknik Elektro. Fakultas Teknik. Universitas Indonesia, Depok:2008.

[5] Yana, Yuli. Rancang Bangun Solar Tracker Berdasarkan Waktu Menggunakan RTC DS 3231. **Tugas Akhir.** Jurusan Teknik Elektro. Politeknik Negeri Sriwijaya. Palembang: 2016.

[6] Kurniawan, Abu Hatim, Gita Dwi Permata Sari. Alat Monitoring Dan Alarm Denyut Jantung Manusia Terintregasi Android Berbasis Atmega 328. **Tugas Akhir.** Program Studi D3 Teknik Elektro. Jurusan Teknik Elektro. Fakultas Teknologi Industri. Institut Teknologi Sepuluh Nopember. Surabaya: 2016.

[7] Wakur, Jansen Silwanus. Alat Penyiram Tanaman Otomatis menggunakan Arduino Uno. **Tugas Akhir.** Jurusan Teknik Elektro. Politeknik Negeri Manado. Manado: 2015.

[8] Salam, Abdul, Mukhidin, Tama Sucita. Rancang Bangun Sistem Jaringan Multidrop Menggunakan RS485 pada Aplikasi Pengontrolan Alat Penerangan Kamar Hotel. **Tugas Akhir**. Universitas Pendidikan Indonesia. Bandung: 2012.

[9] Putra, Eko. **Belajar Mikro Kontroler AT89S51/52/55 Teori dan Aplikasi**. Yogyakarta: Gava Media. 2006.

[10] Maryanto, Dwi Cahya. Pengendali Lampu dan Pintu Garasi dengan Bluetooth Berbasis Mikrokontroler. **Tugas Akhir.** Jurusan Teknik Elektro. Fakultas Sains dan Teknologi. Universitas Sanata Dharma. Yogyakarta: 2016.

[11] Nugroho, Budi Wahyu. Prototype Alat Terapi Autis Berbasis Arduino dengan Variasi Suara Lumba-Lumba Click, Brust dan Whistle. **Tugas Akhir.** Jurusan Teknik Elektro. Fakultas Teknik Universitas Lampung. Bandar Lampung: 2016.

LAMPIRAN A

Progam master clock

```
#include <LiquidCrystal.h>
#include <Wire.h>
#include "RTClib.h"
#include "DFRobotDFPlayerMini.h"

RTC_DS3231 rtc;
char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};

LiquidCrystal lcd(22, 24, 26, 28, 30, 32);

DFRobotDFPlayerMini myDFPlayer;

String DataGps = "";
String GGA = ""; // string DataGps untuk menampung data DataGps dari GPS
String GPRMC = ""; //

// parameter gps
String TimeGps;
String Latitude;
String DirLatitudeNorS;
String Longitude;
String DirLongitudeEorW;
String GpsQuality;

String GPRMC_Time;
String GPRMC_Date;

// string jam
String tampil_jam[60] = {"00", "01", "02", "03", "04", "05", "06", "07", "08", "09", "10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20", "21", "22", "23", "24", "25", "26", "27", "28", "29", "30", "31", "32", "33", "34", "35", "36", "37", "38", "39", "40", "41", "42", "43", "44", "45",
```

```
"46", "47", "48", "49", "50", "51", "52", "53", "54", "55", "56", "57", "58",
"59"};

    int UTC_TimeZone = 7;
    int tanggal, bulan, tahun, jam, menit, detik;
    unsigned long int timerStart = 0, timerStartTampil = 0;
    boolean timeUpdateGpsDurations = 60;//86400; // waktu 24 jam
untuk update (s)
    boolean timeUpdateGpsFlag = 0;

    /*
      jadwal_update_waktu dalam satuan jam 0 - 23
    */
    int jadwal_update_waktu = 23; // 23 pada saat jam 11  tengah malam

    void setup() {
      // put your setup code here, to run once:
      lcd.begin(16, 2);
      pinMode(34, OUTPUT);
      digitalWrite(34, HIGH);
      Serial1.begin(4800); // memulai serial 1 tereima gps
      Serial.begin(9600); // memulai serial 0
      Serial2.begin(9600);
      Serial3.begin(9600);
      //Serial.print("mulai\n");

      delay(3000);
      if (! rtc.begin()) {
        Serial.println("Couldn't find RTC");
        while (1);
      }

      //rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));

      if (rtc.lostPower()) {
        Serial.println("RTC lost power, lets set the time!");
        // following line sets the RTC to the date & time this sketch was
compiled
        //rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
```

```
        // This line sets the RTC with an explicit date & time, for example
to set
        //January 21, 2014 at 3am you would call:
        // rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
       }
        if (!myDFPlayer.begin(Serial3)) {
         while(true);
       }
       myDFPlayer.volume(25);  //Set volume value. From 0 to 30


       // set timer update
       timerStart = millis();
       timerStartTampil = millis();
      }

     // interupsi serial 1
     void serialEvent1() {
       GPS(1);
     }

     void loop() {
       // menagmbil waktu rtc
       DateTime now = rtc.now();
     //  int tahun_rtc = now.year();
     //  int bulan_rtc = now.month();
     //  int tanggal_rtc = now.day();
     //  int hari_rtc = daysOfTheWeek[now.dayOfTheWeek()];
       int jam_rtc = now.hour();
       int menit_rtc = now.minute();
       int detik_rtc = now.second();


       int local_time = 0;

       if (jam_rtc == jadwal_update_waktu && timeUpdateGpsFlag == 0)
{
         local_time = jam + 7;
         // filter untuk jam
```

```
        if (local_time > 23) {
          local_time %= 24;
        }

        // nunggu gps fix
        if (GpsQuality == "1" || GpsQuality == "2") {
          // update
          rtc.adjust(DateTime(tahun, bulan, tanggal, local_time, menit,
detik)); // adjust untuk sinkron waktu dari gps ke rtc


Serial.println("++++++++++++++++++++++++++++++\nwaktu telah
diperbaharui dg gps!\n++++++++++++++++++++++++++++++");
          timeUpdateGpsFlag = 1;
        }
      }
      if (jam_rtc != jadwal_update_waktu) {
        timeUpdateGpsFlag = 0;
      }

      // print ke lcd
      String buff_jam = tampil_jam[jam_rtc] + ":" +
tampil_jam[menit_rtc] + ":" + tampil_jam[detik_rtc];

      if (timerStartTampil + 3000 > millis() ) {
        lcd.setCursor(0, 0);
        lcd.print ("Master Time&Date");
      } else if (timerStartTampil + 6000 > millis()) {
        if (GpsQuality == "1" || GpsQuality == "2") {
          lcd.setCursor(0, 0);
          lcd.print("    GPS Fix!     ");
        }
        if (GpsQuality == "0") {
          lcd.setCursor(0, 0);
          lcd.print("   GPS no Fix!   ");
        }
        //timerStartTampil = millis();
      } else if (timerStartTampil + 9000 > millis()) {
        timerStartTampil = millis();
```

```
  }

  lcd.setCursor(4, 1);
  lcd.print(buff_jam);

  //suara jam
  if(buff_jam == "00:00:00") {myDFPlayer.play(12);}
  if(buff_jam == "01:00:00") {myDFPlayer.play(1);}
  if(buff_jam == "02:00:00") {myDFPlayer.play(2);}
  if(buff_jam == "03:00:00") {myDFPlayer.play(3);}
  if(buff_jam == "04:00:00") {myDFPlayer.play(4);}
  if(buff_jam == "05:00:00") {myDFPlayer.play(5);}
  if(buff_jam == "06:00:00") {myDFPlayer.play(6);}
  if(buff_jam == "07:00:00") {myDFPlayer.play(7);}
  if(buff_jam == "08:00:00") {myDFPlayer.play(8);}
  if(buff_jam == "09:00:00") {myDFPlayer.play(9);}
  if(buff_jam == "10:00:00") {myDFPlayer.play(10);}
  if(buff_jam == "11:00:00") {myDFPlayer.play(11);}
  if(buff_jam == "12:00:00") {myDFPlayer.play(12);}
  if(buff_jam == "13:00:00") {myDFPlayer.play(1);}
  if(buff_jam == "14:00:00") {myDFPlayer.play(2);}
  if(buff_jam == "15:00:00") {myDFPlayer.play(3);}
  if(buff_jam == "16:00:00") {myDFPlayer.play(4);}
  if(buff_jam == "17:00:00") {myDFPlayer.play(5);}
  if(buff_jam == "18:00:00") {myDFPlayer.play(6);}
  if(buff_jam == "19:00:00") {myDFPlayer.play(7);}
  if(buff_jam == "20:00:00") {myDFPlayer.play(8);}
  if(buff_jam == "21:00:00") {myDFPlayer.play(9);}
  if(buff_jam == "22:00:00") {myDFPlayer.play(10);}
  if(buff_jam == "23:00:00") {myDFPlayer.play(11);}

  // kirim ke slave
  Serial2.print(buff_jam);
  Serial2.print("\n");


}

// fungsi GPS
```

```
void GPS(boolean debug) {
  int indexDataGps;
  int indexTime; // 1
  int indexLatitude; // 2
  int indexNorS; // 3
  int indexLongitude; // 4
  int indexEorW; // 5
  int indexGpsQuality; // 6
  int indexNumOfSatellites; // 7
  int indexHorizonDP; // 8
  int indexGeoid; // 9
  int indexUnitOfantenna; // 10
  int indexGeoidalSep; // 11
  int indexUnitGeodial; // 12
  int indexAgeOfDifGps; // 13
  int indexDifRefStaId; // 14
  int indexChecksum; // 15

  /*
    Mengambil data GPS dari serial 1 arduino mega
  */
  while (Serial1.available() > 0) {
    DataGps = Serial1.readStringUntil('\r');
  }

  /*
     Mengambil data $GPDataGps saja
  */
  if (DataGps.substring(0, 7) == "\n$GPGGA") {
    //Serial.println(DataGps);

    /*
      Parshing data $GPDataGps
```

Global Positioning System Fix Data. Time, Position and fix related data for a GPS receiver :

1) Time (UTC)
2) Latitude
3) N or S (North or South)

```
        4) Longitude
        5) E or W (East or West)
        6) GPS Quality Indicator,
        0 - fix not available,
        1 - GPS fix,
        2 - Differential GPS fix
        7) Number of satellites in view, 00 - 12
        8) Horizontal Dilution of precision
        9) Antenna Altitude above/below mean-sea-level (geoid)
        10) Units of antenna altitude, meters
        11) Geoidal separation, the difference between the WGS-84 earth
          ellipsoid and mean-sea-level (geoid), "-" means mean-sea-level
below ellipsoid
        12) Units of geoidal separation, meters
        13) Age of differential GPS data, time in seconds since last
SC104
          type 1 or 9 update, null field when DGPS is not used
        14) Differential reference station ID, 0000-1023
        15) Checksum

        lihat datasheet NMEA0183 halaman 9

      */
      indexDataGps = DataGps.indexOf(',');
      indexTime = DataGps.indexOf(',', indexDataGps + 1); // 1
      indexLatitude = DataGps.indexOf(',', indexTime + 1); // 2
      indexNorS = DataGps.indexOf(',', indexLatitude + 1); // 3
      indexLongitude = DataGps.indexOf(',', indexNorS + 1); // 4
      indexEorW = DataGps.indexOf(',', indexLongitude + 1); // 5
      indexGpsQuality = DataGps.indexOf(',', indexEorW + 1); // 6
      indexNumOfSatellites = DataGps.indexOf(',', indexGpsQuality +
1); // 7
      indexHorizonDP = DataGps.indexOf(',', indexNumOfSatellites +
1); // 8
      indexGeoid = DataGps.indexOf(',', indexHorizonDP + 1); // 9
      indexUnitOfantenna = DataGps.indexOf(',', indexGeoid + 1); // 10
      indexGeoidalSep = DataGps.indexOf(',', indexUnitOfantenna +
1); // 11
```

```
        indexUnitGeodial = DataGps.indexOf(',', indexGeoidalSep + 1); //
12
        indexAgeOfDifGps = DataGps.indexOf(',', indexUnitGeodial +
1); // 13
        indexDifRefStaId = DataGps.indexOf(',', indexAgeOfDifGps + 1);
// 14
        indexChecksum = DataGps.indexOf(',', indexDifRefStaId + 1); //
15

        //String secondValue = myString.substring(Index1 + 1, Index2);
        //String thirdValue = myString.substring(Index2 + 1, Index3);
        TimeGps = DataGps.substring(indexDataGps + 1, indexTime);
        Latitude = DataGps.substring(indexTime + 1, indexLatitude);
        DirLatitudeNorS = DataGps.substring(indexLatitude + 1,
indexNorS);
        Longitude = DataGps.substring(indexNorS + 1, indexLongitude);
        DirLongitudeEorW = DataGps.substring(indexLongitude + 1,
indexEorW);
        GpsQuality = DataGps.substring(indexEorW + 1,
indexGpsQuality);

        if (debug) {
          //Serial.print("\n======== GGA Data Gps ========\n");
          //Serial.print("Time GPS\t:");
          //Serial.println(TimeGps);

          //Serial.print("Latitude\t:");
          //Serial.print(Latitude);
          //Serial.print(" - ");
          //Serial.println(DirLatitudeNorS);
          //Serial.print("Longitude\t:");
          //Serial.print(Longitude);
          //Serial.print(" - ");
          //Serial.println(DirLongitudeEorW);
          //Serial.print("GPS Quality\t:");
          if (GpsQuality == "0") Serial.print("fix not available");
          if (GpsQuality == "1") Serial.print("GPS fix");
          if (GpsQuality == "2") Serial.print("Differential GPS fix");
          Serial.print("\n\n");
```

```
  }
 }

 /*
  data GPRMC
 */
 if (DataGps.substring(0, 7) == "\n$GPRMC") {
  GPRMC = DataGps;

  /*
   1) Time (UTC)
   2) Status, V = Navigation receiver warning
   3) Latitude
   4) N or S
   5) Longitude
   6) E or W
   7) Speed over ground, knots
   8) Track made good, degrees true
   9) Date, ddmmyy
   10) Magnetic Variation, degrees
   11) E or W
   12) Checksum
  */

  int index_GPRMC = GPRMC.indexOf(',');
  int index_time_GPRMC = GPRMC.indexOf(',', index_GPRMC +
1);
  int index_status_GPRMC = GPRMC.indexOf(',',
index_time_GPRMC + 1);
  int index_Latitude_GPRMC = GPRMC.indexOf(',',
index_status_GPRMC + 1);
  int index_NorS_GPRMC = GPRMC.indexOf(',',
index_Latitude_GPRMC + 1);
  int index_Longitude_GPRMC = GPRMC.indexOf(',',
index_NorS_GPRMC + 1);
  int index_EorW_GPRMC = GPRMC.indexOf(',',
index_Longitude_GPRMC + 1);
  int index_Speed_GPRMC = GPRMC.indexOf(',',
index_EorW_GPRMC + 1);
```

```
        int index_Track_GPRMC = GPRMC.indexOf(',',
index_Speed_GPRMC + 1);
        int index_Date_GPRMC = GPRMC.indexOf(',',
index_Track_GPRMC + 1);
        int index_MagneticVar_GPRMC = GPRMC.indexOf(',',
index_Date_GPRMC + 1);
        int index_Mag_EorW_GPRMC = GPRMC.indexOf(',',
index_MagneticVar_GPRMC + 1);
        int index_Checksum_GPRMC = GPRMC.indexOf(',',
index_Mag_EorW_GPRMC + 1);

        GPRMC_Time = GPRMC.substring(index_GPRMC + 1,
index_time_GPRMC );
        GPRMC_Date = GPRMC.substring(index_Track_GPRMC + 1,
index_Date_GPRMC);

        char Date_buff[7];
        GPRMC_Date.toCharArray(Date_buff, 7);
        char day_buff[3] = {Date_buff[0], Date_buff[1]};
        char month_buff[3] = {Date_buff[2], Date_buff[3]};
        char year_buff[3] = {Date_buff[4], Date_buff[5]};
        tanggal = atoi(day_buff);
        bulan = atoi(month_buff);
        tahun = atoi(year_buff);

        char Time_buff[16];
        GPRMC_Time.toCharArray(Time_buff, 16);
        char jam_buff[3] = {Time_buff[0], Time_buff[1]};
        char menit_buff[3] = {Time_buff[2], Time_buff[3]};
        char detik_buff[3] = {Time_buff[4], Time_buff[5]};
        jam = atoi(jam_buff);
        menit = atoi(menit_buff);
        detik = atoi(detik_buff);

        jam += UTC_TimeZone;
        tahun += 2000;

        if (debug) {
          if (GpsQuality == "1" || GpsQuality == "2") {
```

```
      Serial.print("\n======== GPRMC Data Gps ========\n");
      Serial.print("Date\t:");
      Serial.print(tanggal);
      Serial.print("/");
      Serial.print(bulan);
      Serial.print("/");
      Serial.println(tahun);

      Serial.print("Time\t:");
      Serial.print(jam);
      Serial.print(":");
      Serial.print(menit);
      Serial.print(":");
      Serial.println(detik);

    }
    else {
      //Serial.println("GPS Belum Fix!");
    }
    //Serial.print("Time\t:");

    //Serial.println(GPRMC_Hms);
    //time_delay(100);
   }
   //Serial.print(GPRMC);
   GPRMC = "";
  }



}

void time_delay(unsigned long int ms) {
  unsigned long int now = millis();
  while (ms + now > millis());
}
```

Halaman ini sengaja dikosongkan

LAMPIRAN B

Program *slave clock*

```
#include <SoftwareSerial.h>
#include <SPI.h>
#include <DMD2.h>
#include <fonts/ArialBold14.h>
//#include "DFRobotDFPlayerMini.h"

#define txb 3
#define rxb 2



//SoftwareSerial suara(4, 5);// rx tx untuk suara
SoftwareSerial bt(txb, rxb); // rx, rx

//DFRobotDFPlayerMini myDFPlayer;

SoftDMD dmd(2, 1); // DMD controls the entire display
DMD_TextBox box(dmd, 0, 2);  // "box" provides a text box to automatically write to/scroll the display

int cnt_jam, cnt_menit, cnt_detik;
char buff_tampil[20];
boolean flag_start = 0, flag_edit_timer = 0;
unsigned long int start_millis = 0;
unsigned long int timer_tick = 0;
int timeout_bt = 0;

int jeda_cown_down = 1000; // dalam ms

void setup() {
  // put your setup code here, to run once:
  //pinMode(4, OUTPUT);
  //digitalWrite(4, LOW); // mode low untuk menerima serial 485
  Serial.begin(9600);
  bt.begin(4800);
  //suara.begin(9600);
```

```
  dmd.setBrightness(20);
  dmd.selectFont(ArialBold14);
  dmd.begin();

/*if (!myDFPlayer.begin(suara))
  { while(true);
   }
   timer_tick = millis();
   myDFPlayer.volume(25);  //Set volume value. From 0 to 30
   myDFPlayer.play(1);
  }
*/
}
void loop() {
  // put your main code here, to run repeatedly:
  if (bt.available() > 0) {
    timeout_bt = 10;
    String buff = bt.readStringUntil('0');

    Serial.println(buff);
    if (buff == "1") {
      cnt_jam++;
    }
    if (buff == "2") {
      cnt_jam--;
    }
    if (buff == "3") {
      cnt_menit++;
    }
    if (buff == "4") {
      cnt_menit--;
    }
    if (buff == "5") {
      cnt_detik++;
    }
    if (buff == "6") {
      cnt_detik--;
    }
```

```
  if (buff == "7") {
    flag_start = 1;
    start_millis = millis();
  }
  if (buff == "8") {
    flag_start = 0;
    cnt_jam = 0;
    cnt_menit = 0;
    cnt_detik = 0;
    dmd.clearScreen();
    dmd.drawString(5, 2, F("RESET!"));
    sDelay(3000);
  }

  if (cnt_jam < 0) cnt_jam = 24;
  if (cnt_jam > 24) cnt_jam = 0;
  if (cnt_menit < 0) cnt_menit = 59;
  if (cnt_menit > 59) cnt_menit = 0;
  if (cnt_detik < 0) cnt_detik = 59;
  if (cnt_detik > 59) cnt_detik = 0;

  // Serial.println(cnt_jam);
  dmd.clearScreen();
 }

 if (timer_tick + 1000 < millis()) {
  if (!flag_start) timeout_bt--;
  Serial.println(timeout_bt);
  if (timeout_bt > 0) flag_edit_timer = 1;
  if (timeout_bt < 0 ) {
    flag_edit_timer = 0;
    timeout_bt = 0;
  }
 //Play the first mp3
  timer_tick = millis();
 }
}
 if (flag_start) {
  flag_edit_timer = 0;
```

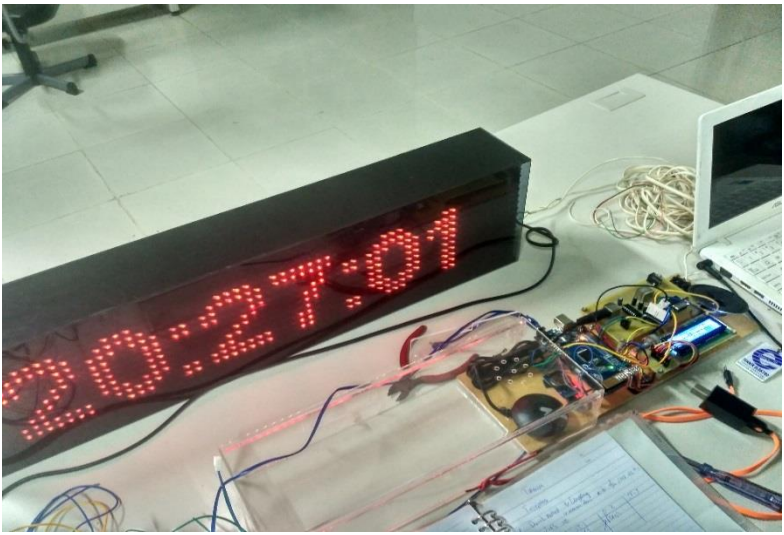
```
        if (start_millis + jeda_cown_down < millis()) {
          cnt_detik--;
          if (cnt_detik < 0) {
            cnt_detik = 59;
            cnt_menit--;
          }
          if (cnt_menit < 0) {
            cnt_menit = 59;
            cnt_jam--;
          }
          if (cnt_jam < 0) {
            cnt_jam = 0;
          }
          //dmd.clearScreen();

          start_millis = millis();

          sprintf(buff_tampil, "%d:%d:%d ", cnt_jam, cnt_menit,
cnt_detik);
          bt.println(buff_tampil);
          dmd.drawString(5, 2, (buff_tampil));

        }


        if (cnt_jam == 0 && cnt_menit == 0 && cnt_detik == 0) {
          flag_start = 0;
          timeout_bt = 0;
          for (int i = 0; i < 3; i++) {
            bt.println("Waktu habis!");
            dmd.clearScreen();
            sDelay(500);
            dmd.drawString(5, 2, F("WAKTU"));
            sDelay(2000);
            dmd.clearScreen();
            sDelay(100);
            dmd.drawString(5, 2, F("HABIS!"));
            sDelay(3000);
          }
```

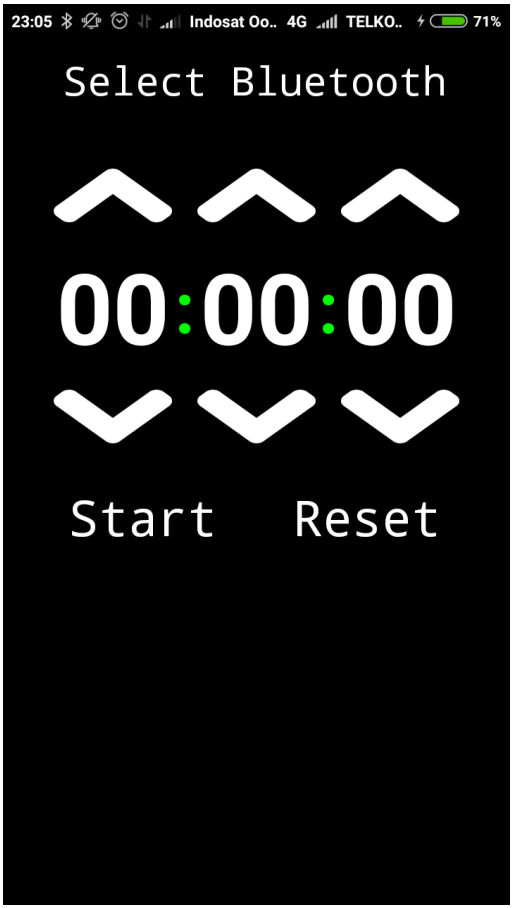
```
          dmd.clearScreen();
        }
      }

      if (flag_edit_timer) {
        sprintf(buff_tampil, "%d:%d:%d ", cnt_jam, cnt_menit,
cnt_detik);
        dmd.drawString(5, 2, (buff_tampil));
      }

      /// program jam serial dari master
      if (!flag_edit_timer && !flag_start) {
        Serial485();
        //dmd.clearScreen();
      }
    }

    void sDelay(unsigned long int jeda) {
      unsigned long int skg = millis();
      while (skg + jeda > millis());
    }

    void Serial485() {
      if (Serial.available() > 0) {
        String dt = Serial.readStringUntil('\n');
        char dtf[25];
        dt.toCharArray(dtf, 25);
        dmd.clearScreen();
        dmd.drawString(5, 2, (dtf));
    //if (dtf == "20:20:00"){myDFPlayer.play(2); }
        //suara mp3
       /* if (dt == "01:00:00") {myDFPlayer.play(1);}
        if (dt == "02:00:00") {myDFPlayer.play(2);}
        if (dt == "03:00:00") {myDFPlayer.play(3);}
        if (dt == "04:00:00") {myDFPlayer.play(4);}
        if (dt == "05:00:00") {myDFPlayer.play(5);}
        if (dt == "06:00:00") {myDFPlayer.play(6);}
        if (dt == "07:00:00") {myDFPlayer.play(7);}
        if (dt == "08:00:00") {myDFPlayer.play(8);}
```

```
    if (dt == "09:00:00") {myDFPlayer.play(9);}
    if (dt == "10:00:00") {myDFPlayer.play(10);}
    if (dt == "11:00:00") {myDFPlayer.play(11);}
    if (dt == "12:00:00") {myDFPlayer.play(12);}
    if (dt == "13:00:00") {myDFPlayer.play(1);}
    if (dt == "14:00:00") {myDFPlayer.play(2);}
    if (dt == "15:00:00") {myDFPlayer.play(3);}
    if (dt == "16:34:00") {myDFPlayer.play(4);}
    if (dt == "17:00:00") {myDFPlayer.play(5);}
    if (dt == "18:00:00") {myDFPlayer.play(6);}
    if (dt == "19:00:00") {myDFPlayer.play(7);}
    if (dt == "20:00:00") {myDFPlayer.play(8);}
    if (dt == "21:00:00") {myDFPlayer.play(9);}
    if (dt == "22:00:00") {myDFPlayer.play(10);}
    if (dt == "23:00:00") {myDFPlayer.play(11);}
    if (dt == "00:00:00") {myDFPlayer.play(12);}
  */}
}
```

LAMPIRAN C

Gambar *master clock* yang terhubung dengan *slave clock*



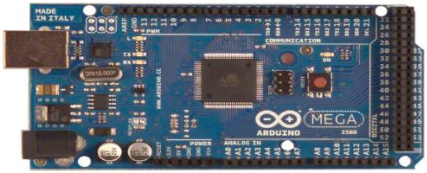

23:05 Indosat Oo.. 4G TELKO.. 71%
Select Bluetooth
00:00:00
Start
Reset

LAMPIRAN D

Datasheet Arduino Mega

ARDUINO

Arduino Mega 2560 Datasheet



Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode() , digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX). Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2). These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
PWM: 0 to 13. Provide 8-bit PWM output with the analogWrite() function.
SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). These pins support SPI communication using the SPI library. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Uno, Duemilanove and Diecimila.
LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH

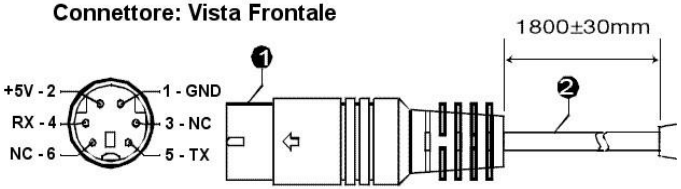
Datasheet GPS BR 355

I.L. ELETTRONICA

GlobalSat
Wired by Wireless

Assegnazione PIN

Connettore: Vista Frontale



PS/2 Connector

Color	Function	CN1
Green	TX	5
White	RX	4
Red	VCC	2
Black	GND	1

❶ Min Din: 6 pin male connector

❷ Wire: 3.6 ± 0.1mm

VCC: l'alimentazione deve essere compresa tra + 4,5 e + 6,5 Vcc, consumo circa 42 mA
TX: dati NMEA provenienti dal Ricevitore GPS, da inviare all'apparato
RX: dati NMEA provenienti da eventuale software esterno (SiRFdemo SOFTWARE)
GND: collegare alla massa (-)

Electrical Characteristics (Receiver)	
Frequency	L1, 1575.42 MHz
C/A Code	1.023 MHz chip rate
Channels	20 channel all-in-view tracking
Sensitivity	-159 dBm
Accuracy	
Position Horizontal	10m 2D RMS (SA off)
Velocity	0.1m/sec 95% (SA off),
Time	1 micro-second synchronized to GPS time
WAAS enabled	5m 2D RMS
Datum	
Datum	WGS-84
Acquisition Rate	
Hot start	1 sec., average (with ephemeris and almanac valid)
Warm start	38 sec., average (with almanac but not ephemeris)
Cold start	42 sec., average (neither almanac nor ephemeris)
Reacquisition	0.1 sec. average (interruption recovery time)
Protocol	
GPS Protocol	Default: NMEA 0183 (Secondary: SiRF binary)
GPS Output Data	SiRF binary >> position, velocity, altitude, status and control ; NMEA 0183 protocol supports command: GGA, GSA, GSV, RMC, VTG, GLL (VTG and GLL are optional)
GPS transfer rate	Software command setting (Default : 4800,n,8,1 for NMEA)
Dynamic Condition	
Acceleration Limit	Less than 4g
Altitude Limit	18,000 meters (60,000 feet) max.
Velocity Limit	515 meters/sec. (1,000 knots) max.
Jerk Limit	20 m/sec**3
Temperature	
Operating	-40°~ 85°C
Storage	-40°~ 85°C
Humidity	Up to 95% non-condensing
Power	
Voltage	4.5V ~ 6.5V
Current	80mA typical (Continuous mode)
Physical Characteristics	
Dimension	53mm diameter , 19.2mm height
Cable Length	65"

Datasheet modul *Bluetooth* HC 05

Specifications

Hardware features

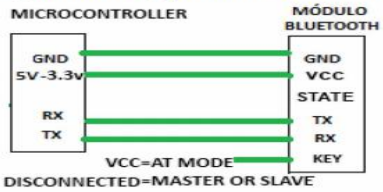
- Typical -80dBm sensitivity.
- Up to +4dBm RF transmit power.
- Low Power 1.8V Operation, 3.3 to 5 V I/O.
- PIO control.
- UART interface with programmable baud rate.
- With integrated antenna.
- With edge connector.

Software features

- Slave default Baud rate: 9600, Data bits:8, Stop bit:1,Parity:No parity.
- PIO9 and PIO8 can be connected to red and blue led separately. When master and slave are paired, red and blue led blinks 1time/2s in interval, while disconnected only blue led blinks 2times/s.
- Auto-connect to the last device on power as default.
- Permit pairing device to connect as default.
- Auto-pairing **PINCODE:"1234"** as default.
- Auto-reconnect in 30 min when disconnected as a result of beyond the range of connection.

Pin out configuration

Typical Application Circuit



Datasheet Arduino Uno

Summary

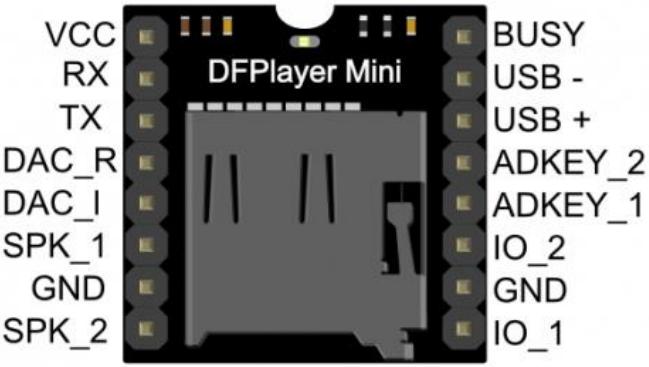
Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V

Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Datasheet DFPlayer Mini

Item	Description
MP3Format	1、Support 11172-3 and ISO13813-3 layer3 audio decoding
	2、Support sampling rate (KHZ):8/11.025/12/16/22.05/24/32/44.1/48
	3、Support Normal、Jazz、Classic、Pop、Rock etc
UART Port	Standard Serial; TTL Level; Baud rate adjustable(default baud rate is 9600)
Working Voltage	DC3.2~5.0V; Type :DC4.2V
Standby Current	20mA
Operating Temperature	-40~+70
Humidity	5% ~95%

Table 2.1 Specification Description



No	Pin	Description	Note
1	VCC	Input Voltage	DC3.2~5.0V;Type: DC4.2V
2	RX	UART serial input	
3	TX	UART serial output	
4	DAC_R	Audio output right channel	Drive earphone and amplifier
5	DAC_L	Audio output left channel	Drive earphone and amplifier
6	SPK2	Speaker-	Drive speaker less than 3W
7	GND	Ground	Power GND
8	SPK1	Speaker+	Drive speaker less than 3W
9	IO1	Trigger port 1	Short press to play previous（long press to decrease volume）
10	GND	Ground	Power GND
11	IO2	Trigger port 2	Short press to play next（long press to increase volume）
12	ADKEY1	AD Port 1	Trigger play first segment
13	ADKEY2	AD Port 2	Trigger play fifth segment
14	USB+	USB+ DP	USB Port
15	USB-	USB- DM	USB Port
16	BUSY	Playing Status	Low means playing \High means no

Table 2.2 Pin Description

RIWAYAT PENULIS

Nama : Candra Mashuri
TTL : Gresik,06 Agustus1995
Jenis kelamin: Laki-laki
Agama : Islam
Alamat :Ngemplak Benjeng Gresik
Telp/HP: 085606104346
Email : candramashuri46@gmail.com

Riwayat Pendidikan :

2002-2008: SDN Munggugebang 1

2008-2011: MTs Negeri Gresik

2011-2014: SMK Negeri 1 Cerme Gresik

2014-2017: D3 Teknik Elektro Otomasi, Fakultas Vokasi Institut Teknologi Sepuluh Nopember

Pengalaman kerja :

1. Praktik Industri di PT Sumber Mas Indah Plywood
2. Kerja Praktek di PT CNC Controller Indonesia

Pengalaman Organisasi :

1. Staff Hubungan Luar (HUBLU) Pengurus Bidik Misi ITS (BIMITS) Periode 2015/2016
2. Anggota Unit Kegiata Mahasiswa (UKM) IAC (ITS Astronomy Club) Periode 2015/2016

Halaman ini sengaja dikosongkan

RIWAYAT PENULIS

Nama	: Samsul Hidayatulloh
TTL	: Jombang, 16 Maret 1996
Jenis kelamin	: Laki-laki
Agama	: Islam
Alamat	:RT04/RW 013 dusun ngembul,desa/Kec.Kesamben, Jombang
Telp/HP	: 081336175663
Email	: samsulhidayat116@gmail.com

Riwayat Pendidikan

2002-2008: SDN Kesamben 1

2008-2011: SMPN 1 Kesamben

2011-2014: SMAM Muhammadiyah 1 Jombang

2014- 2017: D3 Teknik Elektro Otomasi, Fakultas Vokasi Institut Teknologi Sepuluh Nopember

Pengalaman kerja

1. Kerja Praktek di Telkom Indonesia divisi Maintenance Injoko Surabaya
2. Petugas entry data Sensus Ekonomi 2016

Halaman ini sengaja dikosongkan